Fuzzy Logic-Based PID Self-Tuning

by

Taher Mohammed Taher Al-Nemar

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

March, 1996

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

FUZZY LOGIC-BASED PID SELF-TUNING

BY

TAHER MOHAMMED TAHER AL-NEMER

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

MARCH 1996

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

This thesis, written by

TAHER MOHAMMED TAHER AL-NEMER

under the direction of his thesis committee, and approved by all the members, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING.

Thesis Committee

Chairman (Dr. F.M. Al-Sunni)

Member (Dr. M.S. Ahmed)

Member (Dr. L. Cheded)

Department Chairman

Dean, College of Graduate Studies

Date : 24.3.96

Acknowledgments

I thank Allah, the Lord of worlds, for His mercy and limitless help and guidance. May peace and blessings be upon Mohammed the last of the messengers.

Acknowledgment is due to KFUPM for the support of this project.

I would like to express my deep appreciation to my Thesis advisor Dr. F.M. Al-Sunni for his invaluable advice and encouragement. I also wish to thank the other members of my Thesis committee Dr. M.S. Ahmed and Dr. L. Cheded for their helpful suggestions and comments.

My thanks go also to my friends and colleagues, especially Mr. S. Al-Amer, for their support.

I am grateful to my family, especially my wife for her patience and understanding during my busy schedule.

Contents

LIST OF TABLES , i

LIST OF FIGURES , ii

ABSTRACT (ARABIC) , v

ABSTRACT (ENGLISH) , vi

1. INTRODUCTION , 1

1.1 Introduction 1
1.2 Literature Survey.....5
1.3 Thesis Objectives.....8
1.4 Organization of the Thesis.....10

2. PRELIMINARIES ON PID CONTROL AND FLC , 11

2.1 PID Control..... 11
2.2 Adaptive PID Control..... 16
2.3 Classical PID Auto-tuning..... 17
2.3.1 Z-N Step Response Method..... 17
2.3.2 Z-N frequency response Method..... 18
2.3.3 Optimization Techniques..... 24
2.3.4 Pole Placement..... 25
2.3.5 Cross-Correlation Method..... 25
2.4 Preliminaries on FLC..... 26

2.4.1 Fuzzy sets..... 26
2.4.2 Operations..... 28
2.4.3 Fuzzy Logic Control..... 30

3. FL BASED PID SELF-TUNING , 35

3.1 Introduction..... 35
3.2 Proposed Algorithm Structure..... 37
3.3 Adaptation mechanism..... 39
3.4 Fuzzy Control Rules derivation..... 42
3.5 Scaling factors selection..... 44
3.6 Stability..... 50
3.5 Simulations..... 50
3.6 Comparison to other schemes..... 58
3.7 Conclusions..... 74

4. FL-BASED TIME DELAY COMPENSATION , 75

4.1 Introduction..... 75
4.2 Time delay compensation via SP..... 78
4.3 Modeling mechanism..... 79
4.4 Proposed Controller..... 81
4.5 Simulations..... 86
4.6 Conclusions..... 93

5. FL-BASED GAIN SCHEDULING CONTROLLER , 98

5.1 Introduction..... 98
5.2 Gain Scheduling Control..... 99
5.3 Fuzzy Gain Scheduling Control..... 100
5.4 FL-based GS Controller..... 106

5.5 Simulations..... 107
5.6 Conclusions..... 115

6. CONCLUSIONS , 120

6.1 Summary and Conclusions..... 120
6.2 Recommendations for future work..... 110

References , 124

Appendix A , 129

Vita , 157

List of Tables

Table 2.1 : *PID parameters according to Z-N step response method*..... 21
Table 2.2 : *PID parameters according to Z-N frequency response method*..... 23
Table 3.1 : *Fuzzy Control Rules for* α..... 47
Table 3.2 : *Fuzzy Control Rules for* β..... 47
Table 3.3 : *Fuzzy Control Rules for* α *and the path taken by the FLC*..... 49
Table 3.4 : *Summary of simulation results*..... 62
Table 4.1 : *Summary of simulation results*..... 97
Table 5.1 : *Summary of simulation results*..... 119

List of Figures

Figure 2.1 : *Effect of β on set-point and load disturbance resp* 14

Figure 2.2 : *Set-point Filtering* 15

Figure 2.3 : *Z-N step response method* 20

Figure 2.4 : *Astrom-Hagglund relay auto-tuner* 22

Figure 2.5 : *Examples of fuzzy membership functions* 29

Figure 2.6 : *FLC System* 34

Figure 3.1 : *Proposed FL-based PID ST scheme* 38

Figure 3.2 : *A Block diagram of the calculation of α and β* 41

Figure 3.3 : *Membership functions for E, DE, and H* 45

Figure 3.4 : *Step Response* 46

Figure 3.5 : *Simulation results for example 1* 53

Figure 3.6 : *Simulation results for example 1, with sinusoidal noise signal added to measurement* 54

Figure 3.7 : *Simulation results for example 1 with white noise signal added to the measurement* 55

Figure 3.8 : *Simulation results for example 1, with a perturbed process* 56

Figure 3.9 : *Simulation results for example 2* 59

Figure 3.10 : *Simulation results for example 3* 60

Figure 3.11 : *Simulation results of example 4* 61

Figure 3.12 : *He's scheme (Sampling Time = 0.2)* 68

Figure 3.13 : *He's Scheme (Sampling Time = 0.05)* 69

Figure 3.14 : *Root locus of the closed loop system of the process given by*

(3.15) and the PID tuned using He's scheme 70
Figure 3.15 : Our scheme *(Sampling Time = 0.2)* 71
Figure 3.16 : *Effect of coupling the PID parameters* 72
Figure 3.17 73
Figure 4.1 : *Smith Predictor Control* 82
Figure 4.2 : *Alternative SP Configuration* 83
Figure 4.3 : *Reduced SP Block Diagram* 84
Figure 4.4 : *SP FL-Based Controller* 85
Figure 4.5 : *Simulation results for example 1 (Perfect modeling)* 88
Figure 4.6 : *Simulation results for example 1 (perturbed model)* 89
Figure 4.7 : *Simulation results for example 2* 94
Figure 4.8 : *Simulation results for example 3* 95
Figure 4.9 : *Simulation results for example 4* 96
Figure 5.1 : *Gain Scheduling Control* 103
Figure 5.2 : *Membership Functions* 104
Figure 5.3 : *GS vs. FGS* 105
Figure 5.4 : *FL-based GS Controller* 109
Figure 5.5 : *Simulation Results of example 1-1 , a) Process gain , b) Process outputs with GS and FGS* 110
Figure 5.6 : *Simulation results of example 1-1, a) process output and controller gain K_c when using GS b) process output and controller gain K_c when using FGS* 111
Figure 5.7 : *Simulation Results of example 1-2 , a) Process gain , b) Process outputs with GS and FGS* 113
Figure 5.8 : *Simulation results of example 2-1, a) closed loop response, b)* α *and* β 116
Figure 5.9 : *Simulation results of example 2-2, a) closed loop response,*

b) Process gain K_P and controller gain K_c 117
Figure 5.10 : *Simulation results of example 2-3* 118

خلاصة الرسالة

اسم الطالب : طاهر محمد طاهر النمر

عنوان الرسالة : مولف ذاتي للمتحكم التناسبي التكاملي التفاضلي بواسطة المنطق الغامض

التخصص : هندسة نظم

تاريخ الشهادة : مارس ١٩٩٦

نقدم في هذه الأطروحة مولف ذاتي للمتحكم التناسبي التكاملي التفاضلي معتمد على المنطق الغامض. الفكرة الأساسية المبني عليها هذا المولف هو دمج المتحكم التقليدي مع تقنية المنطق الغامض. في هذا المولف المقترح , يقوم المتحكم التقليدي بالتحكم المباشر بالعملية المراد السيطرة عليها, بينما يقوم المنظم الغامض بتعديل عوامل المتحكم التقليدي بشكل مستمر وذلك لتحسين الاداء. سوف نقوم بتوسيع عمل المولف المقترح ليعالج عمليات ذات وقت ميت طويل و ذلك بدمجه مع متنبئ سميث. كما نطور المولف المقترح في هذه الرسالة ليعالج بعض العمليات التي تحتوي على بعض الدوال الغير خطية و ذلك من خلال دمجه مع منظم ذو عوائد مبرمجة.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول و المعادن

الظهران, المملكة العربية السعودية

مارس ١٩٩٦ م

Abstract

Name : Taher Mohammed Taher Al-Nemer

Title : Fuzzy Logic-Based PID Self-Tuning

Major Field : Systems Engineering

Date of Degree : March 1996

A Fuzzy Logic (FL)-based PID self-tuning regulator is presented in this thesis. The basic concept behind the new scheme is to combine the classical PID control with the knowledge-based Fuzzy Logic technology. The PID controller regulates the process, while the FL part adjusts the PID gains. The scheme is also extended to handle processes with long dead times which commonly exist in many processes. This is accomplished by merging the proposed FL based self-tuning scheme with a dead-time compensation algorithm. Finally, the proposed algorithm will be used to improve the performance of some of the loops that contain known non-linearities by combining a Gain Scheduling algorithm with the FL based PID self-tuning scheme.

Master of Science Degree

King Fahd University of Petroleum & Minerals

Dhahran, Saudi Arabia

March 1996

CHAPTER 1

INTRODUCTION

1.1 Introduction

PID controllers are still considered to be the most widely used controllers in industry, in spite of the availability of more advanced control techniques. In any large industrial plant, hundreds of loops may be regulated by PID controllers [8]. This is due to the PID's simple structure, design and use and its robust performance in a wide range of operating conditions. In addition, control and instrument engineers and operators are familiar with the design and operation of PID controllers. The technology of the PID controller has evolved from pneumatic via analog electronics to being microprocessor based. Nowadays, with the advancement in digital systems and microprocessors, whose computational power is continuously increasing, there is a big effort to enhance the performance of the PID controller, by adding to it other control features like auto-tuning, gain scheduling, and adaptation.

The tuning of PID controllers is an important issue, and it is concerned with the best selection of the three PID parameters (i.e., proportional gain, the derivative and integral time constants) so that an acceptable performance of the control loop is established. The tuning is usually done manually and it is very much based on the experience and knowledge of the control or instrument engineer with the process being tuned.

There are several non-parameteric identification techniques for tuning the PID controller that have been developed over the past decades. Among these are the Ziegler-Nichols (Z-N) step response and the Z-N frequency response methods [7]. All of the tuning methods are based on a model, usually a first or a second order linear model with dead time. Therefore, these techniques might completely fail if the process can not be adequately modeled with a first or second order model, as for instance when the process has a large dead time , or in cases where variations in the process parameters take place. Fine tuning based on experience with individual processes is, usually, required to obtain satisfactory performance.

To overcome the above mentioned problems of dealing with the model complexity and uncertainty, control researchers resorted to more advanced techniques such as the self-tuning (ST) adaptive PID control. The basic idea of PID ST schemes is that the parameters of the controller are frequently updated based on fixed structure models that are identified on-line with parametric identification techniques such as the Recursive Least Squares (RLS) algorithm [3]. In this case, the tuning of

the PID parameters is mostly done using the Z-N tuning formulas (or other methods like pole placement techniques). The disadvantage of the PID ST techniques is that they still rely on some model structure for the process to be controlled and, in addition, they require relatively large computational power.

Fuzzy Logic Control (FLC) is an emerging technique in control. It is felt that FL concepts can be quite useful in this area, as they can play the role of the experienced instrument engineer in using knowledge gained about the process through experience to tune the controller. FLC is a knowledge-based control strategy, that relies on Fuzzy Logic (FL) and fuzzy set theory, first introduced by Zadeh [34]. FL is a logic that is close to the human way of thinking and reasoning and provides a means for modeling and dealing with the approximate and inexact nature of the real world. FLC tries to capture experience and intuition in the form of IF-THEN rules from which conclusions are drawn using fuzzy inference. This type of logic is very convenient for describing systems which are too complex or have uncertainty to be successfully described with mathematical models. Although, the stability and robustness results of fuzzy systems are not well established yet, FLC has nevertheless proved to be a valuable tool in industrial applications

In this thesis, we propose a FL-based PID self-tuning scheme to control stable processes with short dead time. The purpose of the scheme is to enhance the control and adaptation ability with respect to plant dynamic changes and external disturbances. The proposed algorithm is basically a combination of the classical PID control and the knowledge-based FL

technology. While the PID will be wholly in charge of regulating the process and generating the control signals, the FL part will be utilized to determine the PID parameters and, consequently, improve the performance of the PID controller.

Dead time dominant processes are common in the process industry. They occur as a result of the presence of distance velocity lags, recycle loops, and the dead time associated with composition analysis [27]. It is well known that for processes with *large* dead time, the performance obtained with the conventional PID controller is quite limited [11]. This is due to the fact that, in the presence of long time delay in the process, the controller tends to over-react to errors since it does not immediately see the effect of its corrective actions. This makes the response sometimes very oscillatory or even unstable. From a frequency-domain perspective, the presence of a time delay adds a phase lag to the feedback loop which affects closed loop stability [27]. Therefore, to get a stable performance, the controller gains must be reduced below the optimum values obtained for dead time free process and, consequently, resulting a sluggish closed loop response.

The proposed FL-based PID self-tuning scheme will be extended to handle processes with large dead times. This will be accomplished by combining the FL scheme with a dead time compensation algorithm. The best known model-based, dead time compensation controller is the Smith Predictor (SP). It requires a model of the process to predict future process output. We envision that the proposed algorithm will be able to

accommodate some of the model uncertainties that might take place when using SP.

Gain Scheduling (GS) is a special case of adaptive control. GS is a special nonlinear feedback control scheme. It is based on a linear regulator whose parameters are changed as a function of operating conditions in a preprogrammed way [2]. It is a very useful technique to compensate for variations in the process parameters that can be predicted from measured operating conditions or *known nonlinearities* of the process. In fact, it is the foremost method for handling parameters variations in flight control system [2]. In addition, it is widely used in the process industry [27,29]. The heat exchanger is a perfect example of a process that can benefit from GS [29].

We propose in this thesis to improve the transition between the different linear controllers by investigating the Fuzzy Gain Scheduling (FGS) method presented by Ling et. al. [22]. The other objective is to refine the performance of the GS controller by applying the FL-based ST scheme at the local controllers' level.

1.2 Literature Survey

A comprehensive work of automatic tuning on PID regulators has been carried out by Astrom and Hagglund [2,5,6,7,8,12]. A recent Ph.D. thesis by Ho [16] gives insights on the various aspects of PID control and expert system-based PID auto-tuner. A good review of the classical PID

design and applications in the process industry can be found in [27] and [29]. Various aspects of adaptive self-tuning regulators can be found in [2,3].

The basic idea of FL was first presented by professor Zadeh [34]. Stimulated by Zadeh's work, Mamdani implemented the first FL controller in the mid-seventies [19,24], which motivated further studies and industrial applications in the area of control such as Sugeno's [30], Pedrycz's [26], Lee's [21], Berenji's [9], Yamakawa's [33], and many others. There are a number of actual FLC industrial applications reported in the literature, such as the cement kiln, water purification process, automatic train operation, and so many other applications conducted on pilot plants. A comprehensive literature survey of FL applications in control is reported in [30].

The application of the FLC concept in tuning PID controllers can be found in [15,17,18,32,35]. Jamshidi [17] presented a case study in which a PI controller gains are adaptively changed via FL. In this scheme, different values of the PID parameters are tabulated and the FL scheme selects the most suitable set of values based on the information collected about the process.

He [15], Tzafestas [32], and Zhao [35] proposed an approach to intelligent PID control based on FL. Their approach assumes that one has available nominal controller parameter settings through some classical tuning technique (e.g., Z-N). Then, using FL, these parameters are varied during system operation so that improved transient and steady state

performance are achieved. In both [15] and [32], some recursive computations are used to update the PID parameters.

In [35], only the proportional gain K_c and the derivative gain K_d are varied using FL inference whereas the integral gain is determined with reference to the derivative time constant. The algorithm in [35], therefore, can be applied for a PI controller instead of a PID. In addition, its is assumed in [35] that K_c and K_d are in prescribed ranges [$K_{c,min}$, $K_{c,max}$] and [$K_{d,min}$, $K_{d,max}$], respectively. $K_{c\,,\,min}$ is taken as half the Z-N value while $K_{c\,,\,max}$ is equal to the Z-N value. On the other hand, $K_{d,\,min}$ is assigned to the Z-N value and $K_{d,max}$ is equal double the Z-N value. The adaptation scheme, hence, starts from the minimum values of the parameters and using FL adaptation the parameters are varied in the prescribed range.

The fuzzy pre-compensated PID controller in [18] is based on the idea of generating a signal which is added to the reference setpoint to form a modified error signal. This error signal is operated on by the PID obtained from Z-N tuning experiment. In other words, it is a fuzzified version of the set-point filtering technique that will be covered in section 2.1, where the filter time constant will be varied using FL.

The idea of dead time compensation was first presented by O. J. Smith (1957) and, nowadays, it is known as the Smith Predictor. Donoghue [10] and Schleck et al. [28] have found that the performance of the SP for set-point changes can be as much as 30% better than a conventional controller based on an ISE (Integral Square Error) performance criteria.

A review of the control design approaches for processes with large time delay is reported in [10]. A good coverage of the SP is presented in [25,27]. In [28], an evaluation of the SP for controlling dead time dominated processes is presented; the SP was evaluated relative to the PID controller. Hang et. al. [13] presented a simple modeling technique that is based on the relay feedback experiment [6,7] and that can be used with the SP. A SP using a simple first order model combined with a PI controller requires five parameters to be determined, namely, the PI controller parameters (K_c and T_i) and the process model parameters (i.e., the static gain, time constant T and the dead time). Hagglund [11] presented a simplified SP scheme in which only three, instead of five, parameters need to be manually adjusted.

A good review of the GS concept and many of its applications are presented by Astrom [2]. Ling et. al. [22] presented a fuzzy-based GS algorithm in which a FL high level controller activates the proper local controllers based on the process conditions. Various applications of GS in the process control are given by Seborg [27] and Shinskey [29].

1.3 Thesis Objectives

In this thesis, we will investigate the possibility of incorporating FL in classical control techniques such as PID auto-tuning, dead-time compensation and gain scheduling in order to improve their performances and to overcome some of problems encountered in their use. The main objectives of the thesis are as follows:

I. Propose a FL-based method for handling problems related to the model uncertainty and sensitivity of the controller to modeling errors. The proposed method is a simpler alternative to the classical adaptive control schemes used in these situations, with possibly some acceptable loss in performance. In this thesis, we shall utilize the power of FLC to enhance the performance of the PID controller through better tuning. This will be achieved by capturing the control engineer's intuition and experience into fuzzy control rules that would take over the tuning function and, consequently, deliver a superior performance over the PID controller tuned using Ziegler-Nichols method. Our scheme is aimed at improving the PID performance over the previous schemes and overcome some of their deficiencies.

II. Extend the scheme proposed in (I) to handle processes with large dead times. The basic concept, here, is to combine the SP scheme with FL so that an improved performance is obtained. The second objective is to investigate the problem of the SP's sensitivity to modeling errors. It is expected that using FL, some of the uncertainties incurred by the model-process mismatch can be accommodated.

III. Combine the FL-based self-tuning scheme proposed in (I) with a GS controller so that the performance of the local controllers is improved. In addition, the application of a rule-based FL higher

level controller will be investigated in order to improve the transition between the different linear controllers.

The various control schemes that are studied in this thesis are evaluated relative to the PID controller tuned with the Z-N frequency response method, as outlined in section 2.3.2. The comparison will be based on the performance characteristics of the step response such as the rise time, overshoot, and settling time and the error performance indices : the Integrated Square Error (ISE) and the Integrated Absolute Error (IAE).

1.4 Organization of the Thesis

The thesis is organized as follows. In Chapter two preliminaries on several topics related to PID control and FLC are covered. The FL-based PID ST scheme is presented in Chapter three. The FL-based time delay compensation controller and the FL-based GS controller are treated in Chapters four and five respectively. Conclusions and recommendations for further work are given in Chapter six. Furthermore, a simulation package based on Matlab and Simulink, that illustrates the different algorithms used throughout this thesis is supplied in Appendix A.

CHAPTER 2

PRELIMINARIES ON PID CONTROL AND FLC

This chapter covers a number of topics related to PID control and FLC. The basic PID controller structure with its different forms is presented in section 2.1. An overview of the adaptive control techniques and the methods used in automatic tuning of PID controllers re given in sections 2.2 and 2.3 respectively. In addition, a short review of the basic FL and FLC concepts is presented in section 2.4. This material will be needed in later chapters of this thesis.

2.1 PID Control

The PID controller can be implemented in either the parallel or the series form [16]. The parallel form is given by the following equation:

$$u(t) = K_c\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right] \qquad (2.1)$$

where K_c, T_i and T_d are the proportional gain, integral time constant and derivative time constant respectively; *u(t)* is the control signal and *e(t)* is error between the reference input and the process output.

The problem with the implementation of the above formula is that tuning for a good load disturbance rejection will often give a set-point response that is too oscillatory. On the other hand, tuning for a good set-point response will give a sluggish load disturbance rejection [8]. Hence, a weighting factor β is placed on the set-point in the proportional term as shown in the following equation :

$$u(t) = K_c\left[(\beta y_r - y) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right] \qquad (2.2)$$

By making β less than one, the set-point response overshoot can be reduced without affecting the load disturbance response [7]. The introduction of β provides a means of adjusting the zeros of the closed-loop transfer function which in turn affects the overshoot [16]. Figure 2.1 illustrates the effect of various values of β on the set-point and load disturbance responses of the second order process with the transfer function :

$$G(s) = \frac{1}{(s+1)(0.5s+1)}$$

Another approach mentioned by Ho [16] to solve the above-mentioned problem is to filter the set-point (see Figure 2.2). Set-point filtering has the same merit as the set-point weighting since the load disturbance response will not be affected as the PID controller settings are not changed. Ho [16] found from extensive simulation studies that the set-point weighting is superior to set-point filtering because the speed of response is much less sacrificed for the same reduction in overshoot.

The discrete time equivalent of equation (2.1) is :

$$u(k) = K_c\left[e(k) + \frac{T_s}{T_i}\sum_{i=1}^{k} e(i) + \frac{T_d}{T_s}\Delta e(k)\right] \qquad (2.3)$$

where T_s is the sampling period for the controller, and $\Delta e(k) = e(k) - e(k-1)$.

The series form of the PID controller is given by :

$$G_c(s) = K_c\left(1 + \frac{1}{sT_i}\right)\left(1 + sT_d\right) \qquad (2.4)$$

The PID controller represented by Equation (2.1) is quite general and the series controller given by Equation (2.4) can always be represented in the form of Equation (2.1). It is however, claimed that the series implementation is sometimes easier to tune manually than the parallel [6]. See [6,8,16] for more details about the various PID implementation forms.

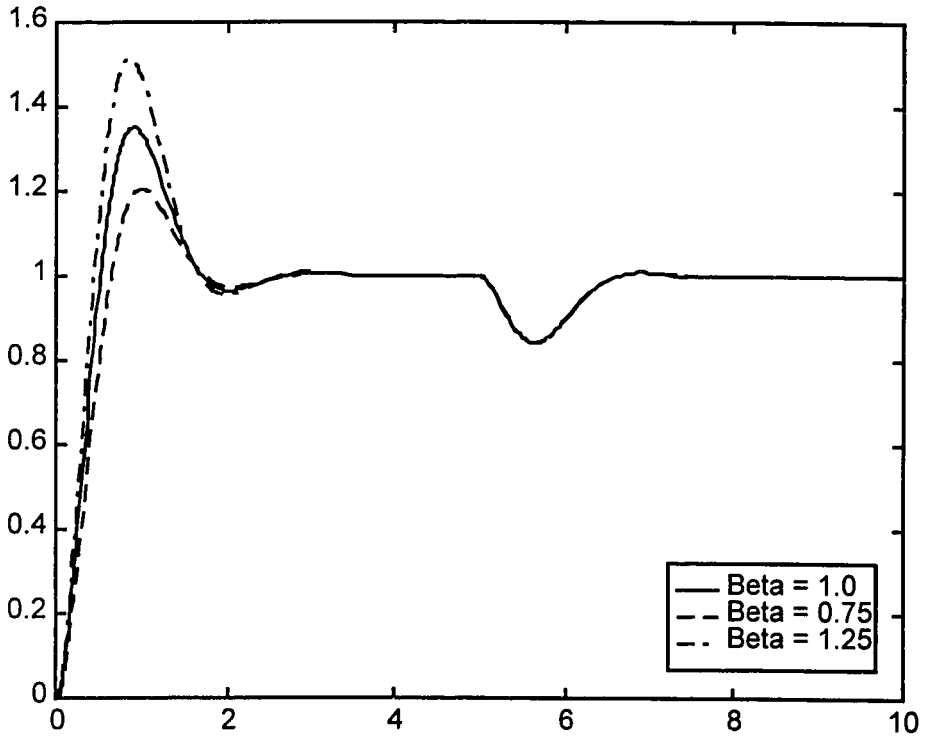


Figure 2.1 : *Effect of β on set-point and load disturbance resp.*

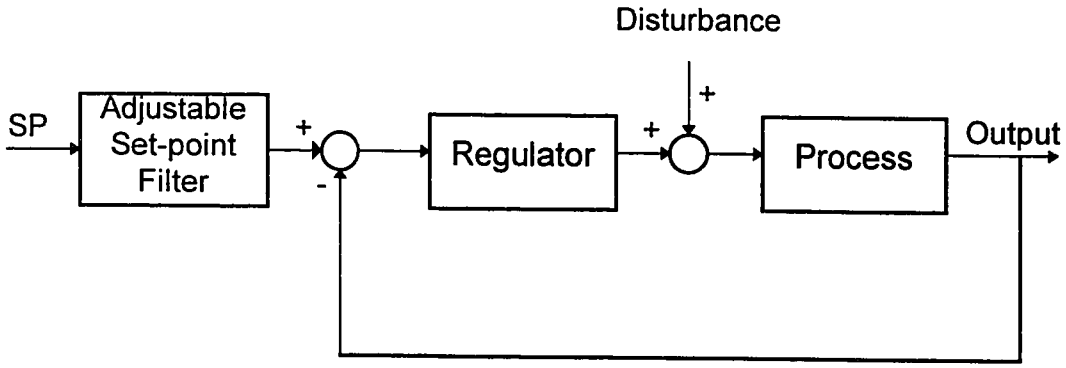


Figure 2.2 : *Set-point Filtering*

2.2 Adaptive PID Control

Adaptive control is a large class of different non-linear control schemes that have the capability to continuously modify their behavior in response to changes in the dynamics of the process and the disturbances. The use of adaptive control techniques usually depends on the dynamics of the process to be controlled. If the process dynamics are constant, a controller with constant parameters should be used and no adaptive control is justified. The parameters of the controller can be obtained using auto-tuning techniques. Auto-tuning is a method where a controller is tuned automatically on demand from the user [8].

If, however, the process dynamics or the nature of disturbances are changing, it is useful to use a controller which can change its parameters to compensate for these variations [8]. Gain scheduling can be used if the variations can be predicted from measured operating conditions, like the variations caused by non-linearities in the control loop. In many cases, however, the variations are not predictable and, consequently, adaptation must be used. Auto-tuning procedures are often used to initialize the adaptation process and this step is called pre-tuning or initial tuning.

2.3 Classical PID Auto-tuning

Auto-tuning is becoming very popular in industry since it can save a lot of engineering time and effort by providing a tool for tuning the wide spread PID controllers [8]. There are many methods for auto-tuning, ranging from heuristic-based to the more sophisticated model-based ones.

2.3.1 Z-N Step Response Method

This method is based on the fact that many processes have a step response, such as one shown in Figure 2.3, and which can be approximated by a first order plus dead time model:

$$G(s) = \frac{K_p}{1+sT} e^{-sL} \qquad (2.5)$$

In this method, the PID regulator's parameters are obtained based on two parameters namely, a and L, that can be directly obtained from the step response as shown in Figure 2.3. The point where the slope of the response has its maximum is first determined, and the tangent at this point is drawn. The intersections of this tangent with the coordinate axes provides the parameters a and L. The Z-N formulas, given in Table 2.1, will roughly give a quarter amplitude damping.

This tuning method gives closed-loop systems that are often too poorly damped and, therefore, finer tuning is usually needed [6]. Another

disadvantage of the method is that it is sensitive to disturbances because it is based on an open loop experiment. There are other design methods that are based on the step response such as the Cohen-Coon method [27]. Next, we talk about another method which avoids the problems encountered by the step response method by implementing a closed loop experiment.

2.3.2 Z-N frequency response Method

This method is based on deriving the PID parameters directly as functions of two parameters: the ultimate gain (K_u) and ultimate period (T_u). These parameters are determined in the original Z-N scheme in the following way. The derivative and integral actions are set to zero and the proportional gain is gradually increased until an oscillation is obtained. The gain at that point is K_u and the period of oscillation is T_u . The ultimate gain and period are actually the description of the intersection point of the Nyquist curve with the negative real axis.

Astrom et. al. [7] proposed an automated way to come up with K_u and T_u (See the feedback loop shown in Figure 2.4). The key idea behind their approach is that large classes of processes will exhibit limit cycle oscillation under relay feedback. The essential process properties, such as the ultimate gain K_u and ultimate period T_u , can be determined from the amplitude and frequency of the limit cycle [7]. The frequency of the limit cycle is approximately the ultimate frequency where the process has a phase lag of 180°. The ratio of the amplitude of the limit cycle and the relay amplitude is approximately the process gain at that frequency. The

period of the oscillation is obtained by measuring the time between zeros crossings and the amplitude is determined by measuring the peak to peak values of the output. Describing function analysis can be used to determine the process characteristics. The describing function of a relay with hysteresis is :

$$N(a) = \frac{4d}{\pi a}\left(\sqrt{1-\left(\frac{\varepsilon}{a}\right)^2} - i\frac{\varepsilon}{a}\right) \qquad (2.6)$$

where d is the relay amplitude, a is the process output amplitude, and ε is the hystersis width [16]. The negative inverse of this describing function is a straight line parallel to the real axis. The oscillation corresponds to the point where the negative inverse describing function crosses the Nyquist curve $G_p(i\omega)$ of the process, where :

$$G_p(i\omega) = -\frac{1}{N(a)} = -\frac{\pi}{4d}\left(\sqrt{a^2-\varepsilon^2} + i\frac{\varepsilon}{a}\right) \qquad (2.7)$$

Now, the ultimate period is taken to be the period of oscillation. The ultimate gain is estimated from the negative reciprocal of the real part of (2.7) [16]:

$$K_u = \frac{4d}{\pi\sqrt{a^2-\varepsilon^2}} \qquad (2.8)$$

Table 2.2 shows the regulator parameters obtained by the Z-N frequency response method. The parameters are given in terms of the ultimate gain K_u and ultimate period T_u that are empirically determined.

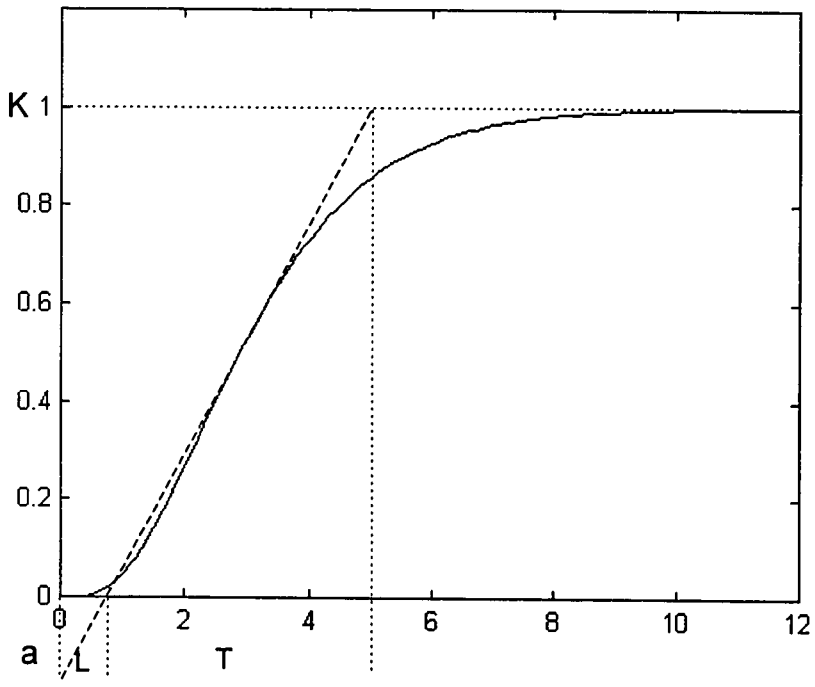


Figure 2.3 : *Z-N step response method*

Table 2.1 : *PID parameters according to Z-N step response method*

Controller	K_c	T_i	T_d
P	$1/a$		
PI	$0.9a$	$3L$	
PID	$1.2/a$	$2L$	$L/2$

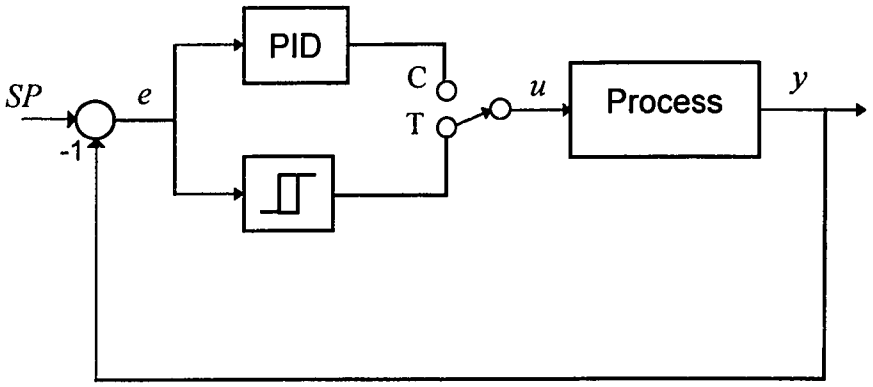


Figure 2.4 : *Astrom-Hagglund relay auto-tuner*

Table 2.2 : *PID parameters according to Z-N frequency response method*

Controller	K_c	T_i	T_d
P	$0.5K_u$		
PI	$0.4K_u$	$0.83T_u$	
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

A block diagram of an autotuner based on the relay method [7] is shown in Figure 2.4. Notice that there is a switch that selects either relay feedback or ordinary PID feedback. When it is desired to tune the system, the PID function is disconnected and the system is connected to relay control. The system then starts to oscillate. The period and the amplitude of the oscillation are determined when steady-state oscillation is obtained. This gives the ultimate period and the ultimate gain.

2.3.3 Optimization Techniques

These are tuning formulas derived to optimize certain criteria such as the Integrated Absolute Error (IAE), the Integrated Time weighted Absolute Error (ITAE), and the Integrated Square Error (ISE). These performance indices are calculated as [27] :

$$IAE = \int_0^\infty |e(t)|dt$$
$$ITAE = \int_0^\infty t|e(t)|dt \qquad (2.9)$$
$$ISE = \int_0^\infty [e(t)]^2 dt$$

Most of these formulas are, however, derived for the first order plus dead time model given by (2.5) and may not give good results for higher order system [16]. Refer to [29] for the tables of formulas of the PID parameters that minimize IAE.

2.3.4 Pole Placement

A complete pole-placement design can be performed if the process is described by a low-order transfer function [6]. Consider the process described by the second-order model in (2.10) :

$$G(s) = \frac{K_p}{(1+sT_1)(1+sT_2)} \tag{2.10}$$

This model has three parameters. By using a PID controller, which also has three parameters, it is possible to arbitrarily place the three poles of the closed loop system [6].

2.3.5 Cross-Correlation Method

In this method, a small pseudo random binary sequence (PRBS) test signal *u(t)* is injected and the resultant process output *y(t)* is logged. The cross correlation $\Phi_{uy}(\tau)$ between *u(t)* and *y(t)* is then used to compute the impulse response of the process as follows [16] :

$$g(\tau) = \frac{1}{A^2 h}\left[\frac{N}{N+1}\right]\left[\Phi_{uy}(\tau) + \sum_{k=0}^{N-1}\Phi_{uy}(k)\right] \tag{2.11}$$

where A is the amplitude of the PRBS signal, h is the sampling interval, N is the length of PRBS, and Nh the period of the PRBS signal. The impulse response is then numerically transformed into its frequency

response from which the ultimate gain, ultimate period, static gain, apparent dead time and time constant of the process can be readily determined [16]. The optimal PID parameters are computed from the ultimate gain and period using the Z-N formula (Table 2.2) [14].

2.4 Preliminaries on FLC

In this section, we give a short review of the basic theory of FL and FLC, that is mostly, based on [21]

2.4.1 Fuzzy sets

FL is based on fuzzy set theory. Fuzzy sets are defined as the sets that do not have a crisply defined membership, but rather allow objects to have grades of membership from 0 to 1 [34]. Fuzzy sets can be viewed as a generalization of the concept of ordinary sets whose membership function only takes two values {0,1}. A fuzzy subset A of a universe of discourse U is defined by a label and a membership function μ_A in which each element u of U receives a number $\mu_A(u)$ in the interval [0,1]. This number represents the grade of membership of u in A. Therefore, a fuzzy set A in U can be represented as a set of ordered pairs of an element u and its grade of membership $\mu_A(u)$:

$$A=\left(u,\mu_A(u)\right),\ u\in U \tag{2.12}$$

Using Fuzzy sets, we can deal with imprecise and vague concepts and data [34]. According to Zadeh [34], the key element in human thinking are not numbers, but tables of fuzzy sets, that is, classes of objects in which the transition from membership to non-membership is gradual rather than abrupt. If we take the example of a speed of a car, for someone speed could be fast, medium or slow. Using FL terms, speed is a linguistic variable that takes the fuzzy sets : fast, medium or slow as its values as shown in Figure 2.5. A speed below 40 is interpreted as slow and a speed above 70 is considered fast. Now, a speed which is between 40 and 55 is medium with certain membership value and slow with another membership value. Similarly, a speed between 55 and 70 can be considered fast with a specified membership value and medium with another membership value.

Fuzzy sets overlap each other which gives the fuzzy inference mechanism the important interpolation capability. Membership functions can be defined either numerically if the universe of discourse is discrete or functionally if the universe of discourse is continuous. Typical membership functions shapes are bell-shaped, triangle-shaped, and trapezoid-shaped functions.

In Fuzzy control applications, for instance, the error (*e*), change of error (*de*), and the change in the controller output (*du*) are often selected as our linguistic variables which can take different fuzzy values, such as positive big, ...etc., and which are determined based on the control application. These linguistic variables are usually related by a set of

fuzzy control rules which cover the whole spectrum of the operation of the process to be controlled. These control rules are of the form

"If e is PB and de is PS, then du is .."

The fuzzy control rules are discussed in more detail in section 2.5.3.

2.4.2 Operations

Assuming that A and B are two fuzzy sets with membership functions of μ_A and μ_B , the following three basic operations can be defined on the sets :

- The *union* of A and B is a fuzzy set with the following membership function :

$$\mu_{A\cup B} = \max\{\mu_A, \mu_B\} \tag{2.13}$$

  This corresponds to the connective "OR"

- The *intersection* of A and B is a fuzzy set:

$$\mu_{A\cap B} = \min\{\mu_A, \mu_B\} \tag{2.14}$$

  This corresponds to the connective "AND"

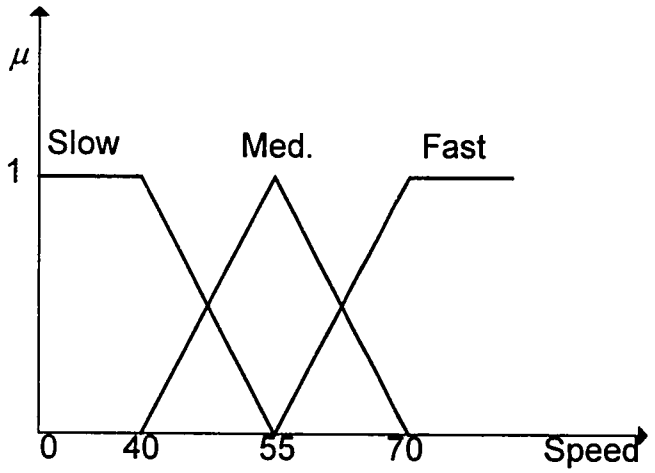


Figure 2.5 : *Examples of fuzzy membership functions*

- The *complement* of a fuzzy set A is a fuzzy set with the following fuzzy set:

$$\mu_{\bar{A}} = 1 - \mu_A \qquad (2.15)$$

This corresponds to the connective "NOT"

The above mentioned operations are most widely accepted as basic fuzzy logic operations. However, there are other operators proposed for intersection, and union such as the T-Norm operators used for intersection and the S-Norm operators used for the union. More details about fuzzy logic operations is given in [21].

2.4.3 Fuzzy Logic Control

The basic configuration of the FLC systems is shown in Figure 2.6. There are four main building block of the FLC : the fuzzification interface, fuzzy control rules base, inference engine, and the de-fuzzification interface.

Fuzzification : is the first operation conducted, and it involves first transforming the range of values of the observed input variables into the corresponding universes of discourse, and then converting the input data within each universe into suitable linguistic values. In other words, the purpose of the fuzzifier is to transform the crisp input space $U \subset R^n$ to a fuzzy set defined in U and characterized by a membership function $\mu_A : U \to [0,1]$ which is labeled by a linguistic term such as 'small',

'big', ...etc. There are a number of known fuzzifier, but the most common one is the singleton fuzzifier, which interprets an input y_o as a fuzzy set A with the membership function $\mu_A(y)$ equal to zero except at the point y_o at which $\mu_A(y_o)$ equals one.

Fuzzy Control Rules base : represents the linguistic information and knowledge base obtained from the expert experience and control engineering knowledge. It is the collection of fuzzy control rules that are expressed as fuzzy conditional statements { IF (Antecedent) THEN (Consequent) }. The antecedent is a fuzzy variable that represent a condition in its application domain and the consequent is another fuzzy variable which is the control action. In general, the fuzzy control rules have the following form :

$$R_j: \text{ if } x \text{ is } A_j, \text{ and } y \text{ is } B_j \text{ then } z \text{ is } C_j , \quad j = 1,2, .., n \qquad (2.16)$$

where x and y are the input variables to the fuzzy system, z is its output variable, and A_j, B_j, and C_j are fuzzy variables with the fuzzy membership functions $\mu_{Aj}(x)$, $\mu_{Bj}(y)$ and $\mu_{Cj}(z)$, respectively.

Each fuzzy control rule can be looked upon as describing a relation between fuzzy variables. This means that a fuzzy conditional statement can be implemented by a fuzzy implication (fuzzy relation) as $A_j \times B_j \rightarrow C_j$, which is a fuzzy set defined in the product space $U \times R_j$. Many fuzzy implication rules have been proposed in the literature, but the most commonly used one, and the one we have used in this thesis, is the minimum operation rule which is defined as follows :

$$\mu_{R_j}(x,y,z) = \mu_{A_j \times B_j \to C_j}(x,y,z) = \min\left[\mu_{A_j}(x), \mu_{B_j}(y), \mu_{C_j}(z)\right] \quad (2.17)$$

Inference Engine : is a mapping from fuzzy sets in U to fuzzy sets in R, based on the fuzzy control rules and the compositional rule of inference. Let A_x be an arbitrary fuzzy set in U, then each R_j determines a fuzzy set (A_x o R_j) in R based on the following compositional rule of inference :

$$m_{A_x \circ R_j}(z) = \max \ \min\left[m_{A_x}(x,y), m_{R_j}(x,y,z)\right] \quad (2.18)$$

The control rules are connected by the connective *else* which is interpreted as the *or* condition. The most common operator used for the fuzzy set *or* operation, is the *max.* operator. Therefore, the overall result for all fuzzy control rules is computed by combining individual results from each rule by the max. operator.

De-Fuzzification : involves converting the inferred fuzzy control action into a crisp value and then scaling that crisp control action into the range suitable for the final control element. The most commonly used method is the one known as the *center of Area (COA)* method. In this method, the centroid under the resulting fuzzy membership function obtained from the inference process is calculated. In the case of a discrete universe, COA gives :

$$z = \frac{\sum_{i=1}^{n} \mu_j(w_i) \cdot w_i}{\sum_{i=1}^{n} \mu_j(w_i)} \tag{2.19}$$

Where n is the number of quantization levels of the output.

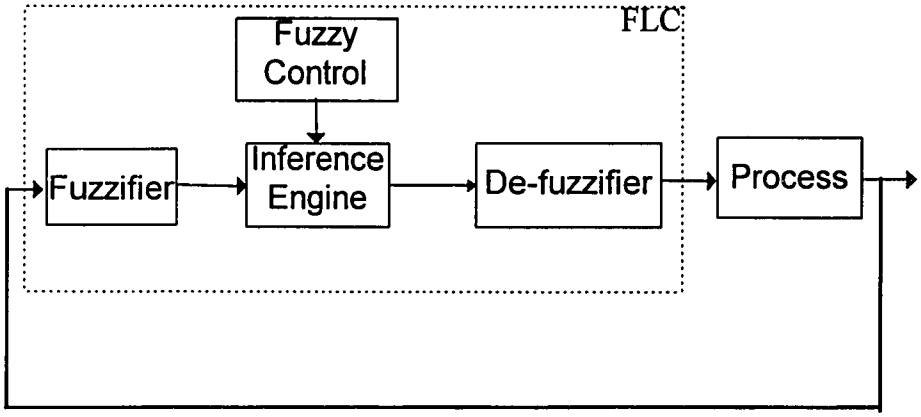


Figure 2.6 : *FLC System*

CHAPTER 3

FL-BASED PID SELF-TUNING

In this chapter, a detailed presentation of the proposed FL-based adaptation scheme is given.

3.1 Introduction

The objective of the FL-based PID ST scheme is to improve the performance of PID controllers by using FL rules. It is expected that considerable improvement in the controller's performance, in reducing the overshoot and speeding up the system's response, can be obtained through small changes in the values of the three PID coefficients using a fuzzy matrix that contains the experience of a human controller. The fuzzy matrix is nothing but the linguistic descriptions of human expertise in controlling a process which are represented as a set of fuzzy control rules. The scheme is a combination of the principle of PID control and FLC.

The proposed algorithm evaluates the process output at specified time instances and generates appropriate PID tuning parameters instantaneously, based on a number of fuzzy control rules derived from studying the general performance of the process under PID control. The well known Z-N formulas are used to get the initial PID parameters which will be varied, as time evolves, based on the response of the process. The Z-N parameters are obtained using Astrom and Hagglund's relay auto-tuner [6], as outlined in section 2.3.2. The variation of the PID parameters is based on two parameters α and β, with the first one parameterizing the integral and derivative time constants and the second one parameterizing the proportional gain. The values of α and β are obtained from FL adaptation.

This chapter is organized as follows. The proposed algorithm structure is outlined in section 3.2. The adaptation mechanism applied in the scheme and the derivation of the fuzzy control rules are given is sections 3.3 and 3.4 respectively. An overview of the way by which the scaling factors of the FL-based ST scheme are selected is covered in section 3.5. In section 3.6, the performance of the new scheme is compared to the PID controller tuned by the Z-N technique through a series of simulations of different plant models. Finally, a comparison of the proposed algorithm to other similar schemes is conducted in section 3.7.

3.2 Proposed Algorithm Structure

The general structure of the proposed FL-based adaptation scheme is shown in Figure 3.1. It has the basic PID structure, but the PID controller gains are generated by fuzzy inference. The PID controller used in this thesis has the model given by Equation (2.2).

The basic concept of the proposed scheme is to vary the contribution of the proportional, integral and derivative parts of the PID controller, during both the transient and load disturbance periods, so as to get a good performance. As clearly shown in Figure 3.1, the FL ST scheme consists of two main blocks. The first one calculates the parameters α and β based on the error *e(k)* and first difference of the error $\Delta e(k)$, which are obtained according to the following formulas :

$$e(k) = y_r(k) - y(k) \tag{3.1}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{3.2}$$

Both α and β are used to update the PID parameters, in the second block according to some parameterized Z-N equations as we will see in the next section. The purpose of the FL adaptation, here, is to generate appropriate values for α and β at the sampling instances such that when the process output is approaching the setpoint, the PID parameters should speed up that convergence, and when the output starts to deviate from the setpoint, the PID parameters should slow that deviation down.

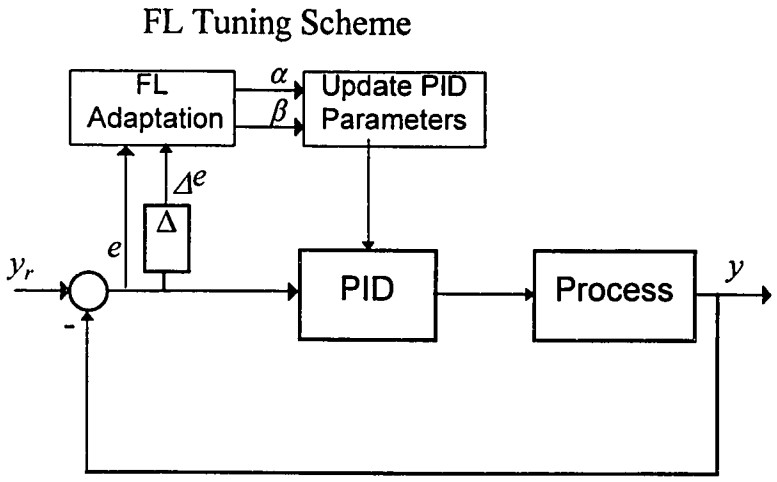


Figure 3.1 : *Proposed FL-based PID ST scheme*

3.3 Adaptation mechanism

The Z-N equations are parameterized with the parameter α so that T_i and T_d are varied according to the following :

$$K_c = 0.6\,K_u \;,\quad T_i = 0.75\,\frac{1}{1+\alpha}\,T_u \;,\quad T_d = 0.25\,T_i \qquad (3.3)$$

Where T_u is the ultimate period and K_u is the ultimate gain which are determined from a relay auto-tuner. We note that for α = 0.5, the Equations in (3.3) reduce to the Z-N formulas given in Table 2.2. The FL adaptation part of the scheme generates appropriate values for α at the sampling instances so that we speed up the convergence of the response to the setpoint and slow down divergence from it.

The other parameter used to update the PID controller is β. This parameter is used to vary the contribution of the proportional term. As can be seen from Equation (2.2), a weighting factor β is placed on the setpoint. The effect of various values of β is as presented in section 2.1 and illustrated by Figure 2.1. The response to setpoint changes depends on the value of β. The controller obtained for different values of β will, however, respond to load disturbances in the same way [6]. The transient response can be considerably improved by varying β during the transient period. Therefore, we will make use of this fact, and will use FL inference mechanism to vary β in such a way to improve the performance of the PID controller. Hence, the FL adaptation part of the

FL ST algorithm will calculate the proper values for β at the sampling instances using fuzzy inference.

Figure 3.2 gives an overview of the calculation of α and β. The error *e(k)* and the first difference of the error *Δe(k)* , as calculated by Equations (3.1)-(3.2), are the inputs to the FL scheme. *sfe*, *sfde*, *sfα* and *sfβ* are scaling factors for *e(k)* , *Δe(k)* , α and β respectively, that need to be adjusted. In general, the selection of appropriate scaling factors is done by trial and error. Once *e(k)* and *Δe(k)* are read by the scheme, they are fuzzified into two linguistic variables E and DE. These fuzzy variables are fed to the inference engine which uses the fuzzy rules, that will be derived in section 3.4, and the fuzzy compositional rule of inference, presented in section 2.4.3, to come up with the proper fuzzy output values for H_α and H_β . Using the center of gravity method, presented in section 2.4.3 , we get crisp values for $h_\alpha(t)$ and $h_\beta(t)$ which are used to update α and β according to the following equations :

$$\alpha(t) = 0.5 + h_\alpha(t) * sf\alpha \quad (3.4)$$

$$\beta(t) = 1.0 + h_\beta(t) * sf\beta \quad (3.5)$$

Equation (3.4) indicates that α will fluctuate around the value 0.5 and, consequently, the integral and derivative parameters will fluctuate around the Z-N values. Similarly, Equation (3.5) indicates that the parameter β will fluctuate around the value 1. This means that the FL scheme will perturb the parameters α and β around the nominal values 0.5 and 1, respectively, so that an improved performance is obtained.

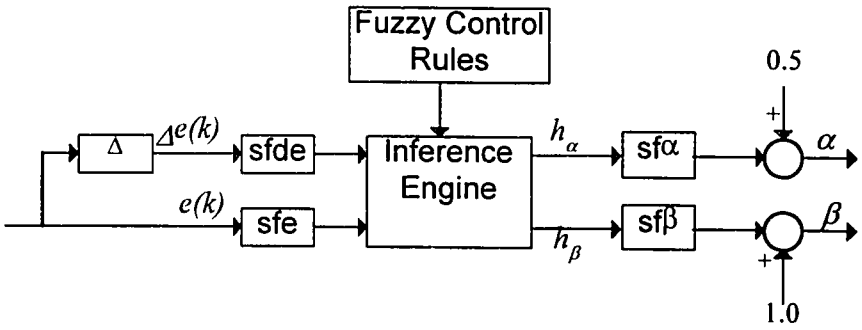


Figure 3.2 : *A Block diagram of the calculation of α and β*

3.4 Fuzzy Control Rules derivation

It is clear from Equations (3.3) that when α increases both T_i and T_d decreases and vice versa. Therefore, it is required that α should increase when the process output is trying to approach the setpoint and, consequently, speeding up the convergence. Similarly, α should decrease when the output is diverging from the setpoint. This causes the deviation to slow down.

The logic we have used to come-up with the fuzzy control rules for β is as follows. If we choose $\beta > 1$ in the early stages of the transient response, we will speed-up the convergence of the response towards the setpoint and, therefore, reduce the rise time. In order not to have a large overshoot, we need to start reducing β before the response approaches the setpoint. As the response deviates from the setpoint in an increasing way, we then need to reduce β even further ($\beta < 1$) to prevent overshoot, and so on.

Basic reasoning and logic just as the above-mentioned can be converted into fuzzy control rules that will be used in the FL adaptation to generate different values for α and β at different stages of the response. The fuzzy control rules have, in general, the following form:

if $e(k)$ *is* E_i *and* $\Delta e(k)$ *is* DE_i *, then* H_α *is* $H1_i$ *and* H_β *is* $H2_i$ $\quad i=1,2, \ldots ,n$ (3.6)

where E_i , DE_i , $H1_i$ and $H2_i$ are fuzzy sets representing the fuzzy variables of the error, change of error, α and β respectively. Each of the error $e(k)$, change of error $\Delta e(k)$, α and β are covered by seven fuzzy sets. The seven sets are: Positive Big (PB), Positive Medium (PM), Positive Small (PS), Zero (Z), Negative Small (NS), Negative Medium (NM), and Negative Big (NB). In this thesis, we have used fuzzy membership functions of the triangular type for E, DE, and the outputs H_1 and H_2 , as shown in Figure 3.3.

The fuzzy control rules of the proposed algorithm have been derived experimentally from studying the step response of the process to be controlled. These rules are summarized in the fuzzy matrices (Table 3.1 and Table 3.2). Table 3.1 gives the fuzzy control rules for α and Table 3.2 gives the rules for β. Figure 3.4 shows an example of a step response.

To illustrate the process by which we have derived the rules, we divided the response in Figure 3.4 into four stages defined by the signs of $e(k)$ and $\Delta e(k)$. In the first stage ($e(k) > 0$, $\Delta e(k) < 0$), i.e. between point (a) and (b), α should be increased first by large values to speed-up the system response and then gradually decreased as $e(k)$ approaches zero so as to avoid overshoot. Similarly, β should be increased in a similar fashion and decreased gradually as the response approaches the setpoint.

In the second stage ($e(k) < 0$, $\Delta e(k) < 0$), i.e. between point (b) and (c), α should be decreased below 0.5 to slow down the divergence from the

setpoint and to avoid overshoot. Similarly, β should be decreased below 1. The magnitude of the decrease of both α and β depends on the value of $\Delta e(k)$. In a similar manner, the logic can be established for the third and fourth stage.

Now, let's look at the fuzzy control rule at point (a). Here, we need a large control signal to achieve a short rise time. Hence, we need large proportional and integral gains and a small derivative gain. This means that α has to have a fuzzy value of PB and β should also take a fuzzy value of PB. Therefore, the fuzzy control rule in this case becomes :

$$\textit{if } e(k) \textit{ is PB and } \Delta e(k) \textit{ is Z , then } H_\alpha \textit{ is PB and } H_\beta \textit{ is PB} \qquad (3.7)$$

The other fuzzy control rules for α and β , that are tabulated in fuzzy tables 3.1 and 3.2 respectively, are derived in a similar way.

3.5 Scaling Factors selection

Once the fuzzy control rules and the forms of membership functions have been determined, the scaling factors need to be tuned. The four scaling factors of membership functions, especially *sfe* and *sfde*, affect the performance of the controller to a large extent. In general, the selection of the appropriate scaling factors is done by trial and error, which is time consuming. This could be considered to be the main drawback of the proposed FL-based ST scheme and FLC in general.

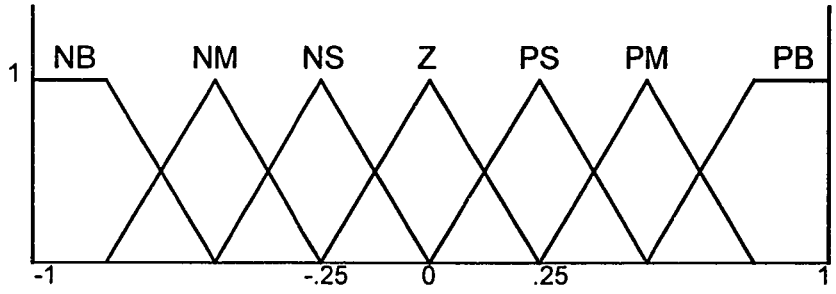


Figure 3.3 : *Membership functions for E, DE, and H*

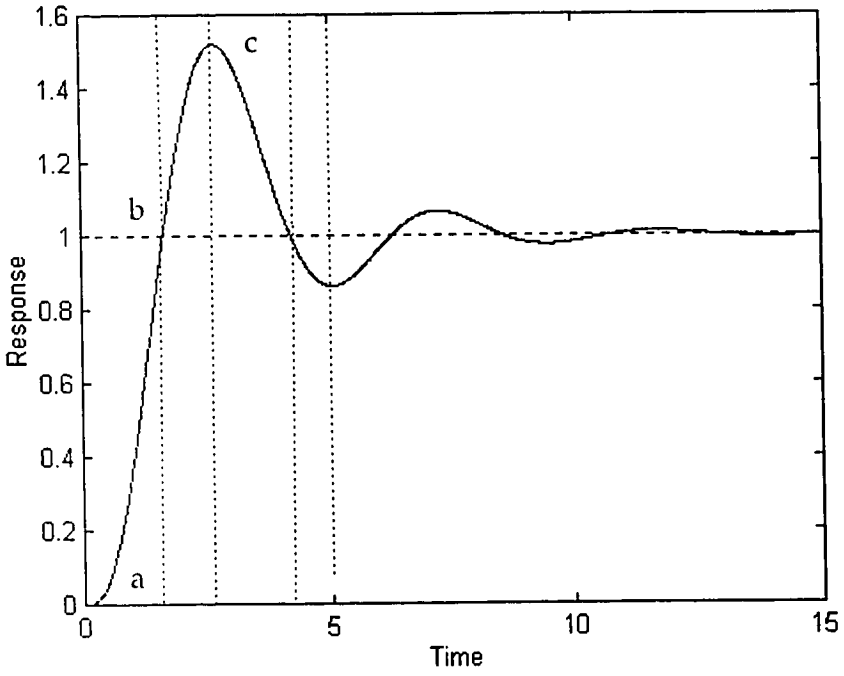


Figure 3.4 : *Step Response*

Table 3.1 : *Fuzzy Control Rules for α*

		DE						
	H1	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
	NB	NB	NB	NM	PB	PM	PS	Z
	NM	NB	NM	NS	PM	PS	Z	NS
E	**NS**	NB	NM	NS	PS	Z	NS	NM
	Z	NB	NM	NS	Z	NS	NM	NB
	PS	NM	NS	Z	PS	NS	NM	NB
	PM	NS	Z	PS	PM	NS	NM	NB
	PB	Z	PS	PM	PB	NM	NB	NB

Table 3.2 : *Fuzzy Control Rules for β*

		DE						
	H2	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
	NB	NB	NB	NB	NB	NM	NS	Z
	NM	NB	NB	NM	NM	NS	Z	PS
E	**NS**	NB	NM	NS	NS	Z	PS	PM
	Z	NB	NM	NS	Z	PS	PM	PB
	PS	NM	NS	Z	PS	PS	PM	PB
	PM	NS	Z	PS	PM	PM	PB	PB
	PB	Z	PS	PM	PB	PB	PB	PB

The main objective of the scaling factors is to convert the input values (i.e., e and Δe) and the output values (i.e., α and β) of the FL ST scheme into their membership functions range. In our application, the membership functions range is between -1 to 1. The effect of the *sfe* and *sfde* is more felt on the performance than the effect of the other two scaling factors . The scaling factors $sf\alpha$ and $sf\beta$ affect only the output of the FL ST scheme and function purely as an overall gain factor that varies the magnitude of the values of α and β.

The effect of the *sfe* and *sfde* can be best explained by looking at Table 3.3, which shows the normal path that the FLC might take when it is in operation. This path will be usually followed when the *sfe* and *sfde* are selected in such a way so as to scale e and Δe into their respective membership functions range. The value of *sfe* and *sfde* have a big effect on that path and, consequently, the performance of the controller. For instance, for some of the processes with fast transient dynamics, it would be better to chose a relatively small *sfe*. This causes the FLC to start at an error (E) with a fuzzy value of PM rather than PB as is shown in Table 3.3. This, in turn, has the effect of producing PID parameters that are not very aggressive initially which contribute to less overshoot. This, however, makes the FL ST scheme less sensitive to the errors around the setpoint, which might results in some oscillations around the setpoint or even an error offset. Selecting an appropriate set of scaling factors is, therefore, problem specific and is very much based on the dynamics of the process considered and the amount of knowledge available about it.

Table 3.3 : *Fuzzy Control Rules for α and the path taken by the FLC*

	DE						
H1	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
NB	NB	NB	NM	PB	PM	PS	Z
NM	NB	NM	NS	PM	PS	Z	NS
NS	NB	NM	NS	PS	Z	NS	NM
Z	NB	NM	NS	Z	NS	NM	NB
PS	NM	NS	Z	PS	NS	NM	NB
PM	NS	Z	PS	PM	NS	NM	NB
PB	Z	PS	PM	PB	NM	NB	NB

E

3.6 Stability

Incorporating the FL inference capability into the conventional control strategies considered in this thesis enhances the performance and the flexibility of these controllers, but at the expense of adding some complexity and nonlinearity into the system. This makes the analysis of the stability of the closed-loop control system very difficult to carry out. One thing that could be done, here, is to add a higher level entity that monitors the performance of the control system and detect instability in an early stage. Once instability is identified, certain corrective action can be taken. For instance, the controller parameters are switched to a set of known stabilizing parameters that guarantee that the control system will remain stable.

3.7 Simulations

Simulations of several plant models have been conducted to study the new algorithm. The setpoint response and the load disturbance response of the new algorithm for each of the simulated models have been compared with their corresponding responses obtained from the PID controller tuned by the Z-N technique.

Example 1

The closed loop response and the trends of α and β for the following model are both shown in Figure 3.5.

$$G_1(s) = \frac{1}{(s+1)(0.5s+1)} \qquad (3.8)$$

The tuning parameters used in this example are as follows :

T_u = 1.261, K_u = 16.1994, *sfe* = 0.67, *sfde* = 0.67, sfα = 0.67; *sfβ* = 0.67

Figure 3.5 illustrates the improvement in performance obtained from the FL-based scheme in terms of rise time, overshoot, and settling time of the setpoint response and load disturbance response. In addition, we have almost zero undershoot in the setpoint response.

This model was also simulated in the presence of noise in the feedback loop. The simulation results are shown in Figures 3.6 and 3.7. The measurement signal was perturbed with a sinusoidal noise signal of amplitude of 0.05 and frequency of 2 rad/s ,as shown in Figure 3.6. One can observe from Figure 3.6 that, compared to the Z-N counterpart, the proposed scheme provides a marked improvement in reducing the effect of noise on the system's response. Figure 3.7 shows the simulation results when a white noise (noise power = 0.005 and sampling period = 0.1) was injected into the measurement signal, and here again the FL-based ST scheme's performance was quite superior to the Z-N one as it provided a considerable reduction in the effect of noise on the system's response.

The control scheme designed for the above example has been simulated for the process with the assumption that the plant dynamics has changed to the following :

$$G(s) = \frac{1}{(1.2s+1)(0.7s+1)} \qquad (3.9)$$

The simulation results shown in Figure 3.8 reveal the robustness of the FL scheme against perturbation in system dynamics.

Example 2

The second example is the following third order system :

$$G_2(s) = \frac{1}{(s+1)^3} \qquad (3.10)$$

with the following tuning parameters :

T_u = 3.726, K_u = 7.5641, *sfe* = 1, *sfde* =1, sfα = 0.67, *sf*β = 0.25

The closed loop response is shown in Figure 3.9. It is clear from this Figure that the FL-based controller outperforms the controller tuned by Z-N in both setpoint and load disturbance responses. The responses of the proposed algorithm have smaller overshoot and settling time, slightly shorter rise time, and the undershoot is almost zero. In addition, the speed of recovery from load disturbance of the FL-based scheme was clearly better than the corresponding Z-N one.

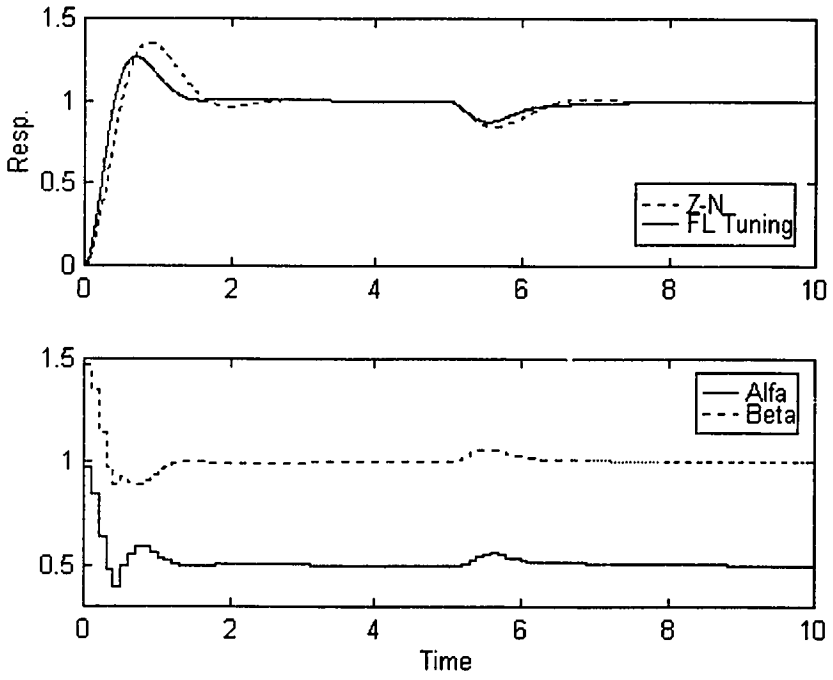


Figure 3.5 : *Simulation results for example 1*

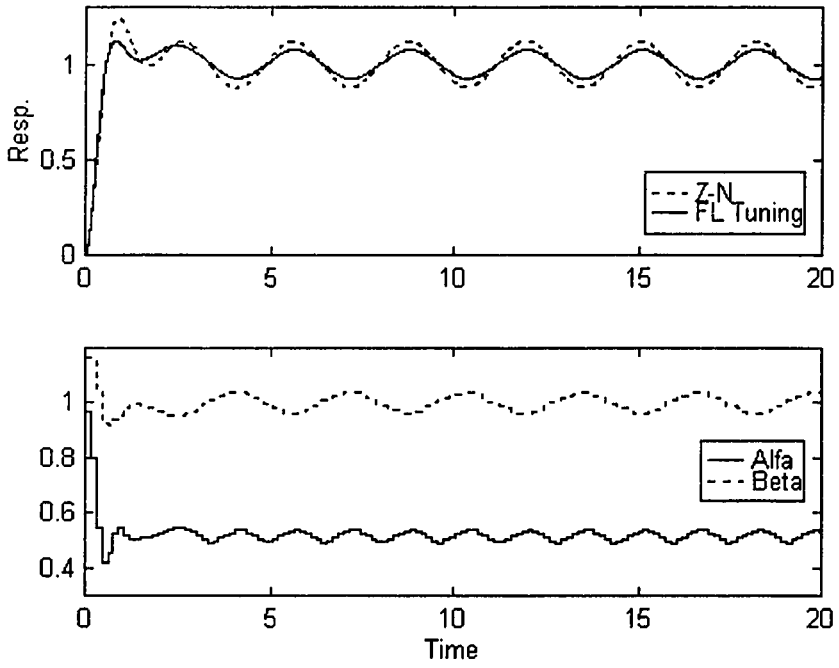


Figure 3.6 : *Simulation results for example 1, with sinusoidal noise signal injected into the feedback*

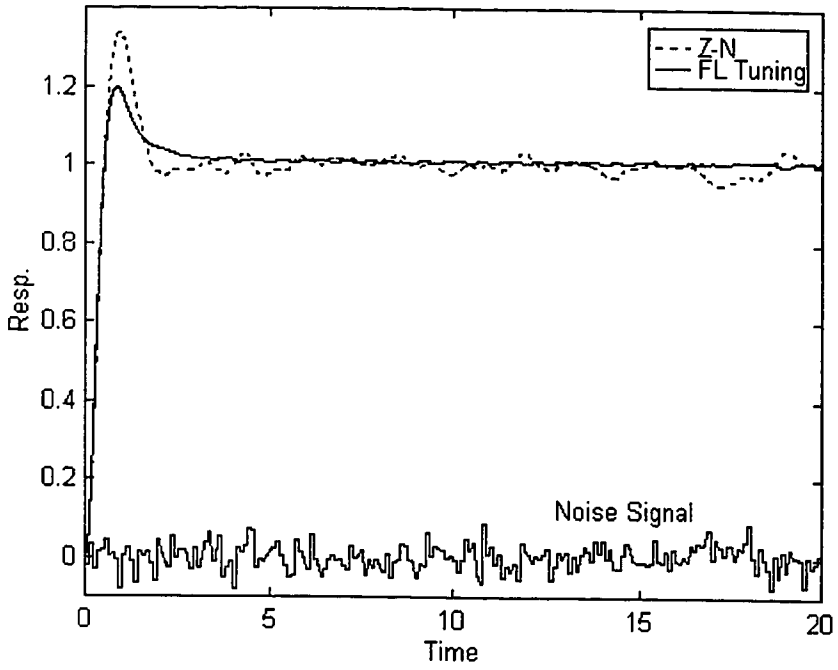


Figure 3.7 : *Simulation results of example 1 with white noise signal added to the measurement*

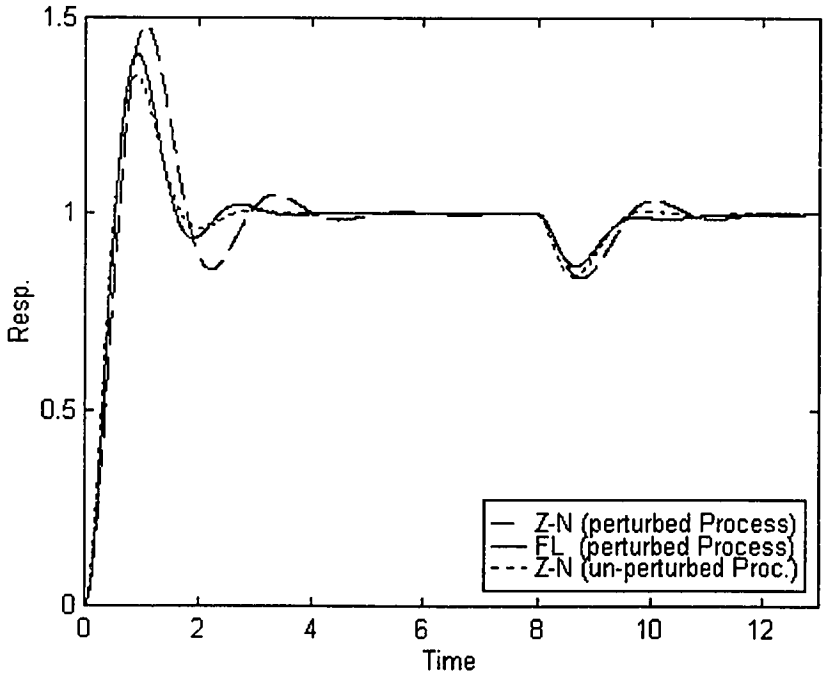


Figure 3.8 : *Simulation results for example 1, with a perturbed process*

Example 3

The following second-order process with small dead time was simulated :

$$G_3(s) = \frac{e^{-0.4s}}{(s+1)^2} \qquad (3.11)$$

The tuning parameters used are :

T_u = 2.88, K_u = 5.7167, *sfe* = 1.0, *sfde* = 1.0, sfα = 0.5, *sfβ* =0.25

The results of the simulation are shown in Figure 3.10. The FL scheme provides a significant improvement in overshoot, slightly shorter rise time, and quite a comparable results to the Z-N in terms of the settling time. The speed of recovery from load disturbance was also improved with the new scheme. The load disturbance response settled at about the same time as the Z-N.

Example 4

The final system considered is the following third order model with dead time :

$$G_4(s) = \frac{e^{-0.4s}}{(s+1)^3} \qquad (3.12)$$

The following tuning parameters are used :

T_u = 4.8235, K_u = 3.965, *sfe* = 1.0, *sfde* = 0.75, sfα = 0.67, *sfβ* = 0.34

The simulation results are shown in Figure 3.11, and they are very similar to the ones obtained from example 3.

The simulations results of the above examples are summarized in Table 3.4, where we have compared their performances in terms of the IAE and ISE criteria.

3.8 Comparison to other schemes

In this section, we mention the major differences between the proposed FL-based ST scheme presented in this chapter and other similar schemes reported in the literature. In particular, we consider, with some detail, the scheme presented by He et. al. [15]. Through simulations, we pinpoint the major differences between the two algorithms. Then, we briefly, discuss the algorithm put forward by Zhao [35] and conduct a simulation example for comparison purposes.

The parameterized Z-N equations used in [15] are as follows :

$$K_c = 1.2\alpha K_u \ , \quad T_i = 0.75 \frac{1}{1+\alpha} T_u \ , \quad T_d = 0.25\, T_i \qquad (3.13)$$

and the following recursive equation is used to update α :

$$\alpha(k+1) = \begin{cases} \alpha(k) + \left(sf\alpha * h_\alpha(k)\right)(1-\alpha(k)) & \text{for } \alpha(k) > 0.5 \\ \alpha(k) + \left(sf\alpha * h_\alpha(k)\right)\alpha(k) & \text{for } \alpha(k) \le 0.5 \end{cases} \qquad (3.14)$$

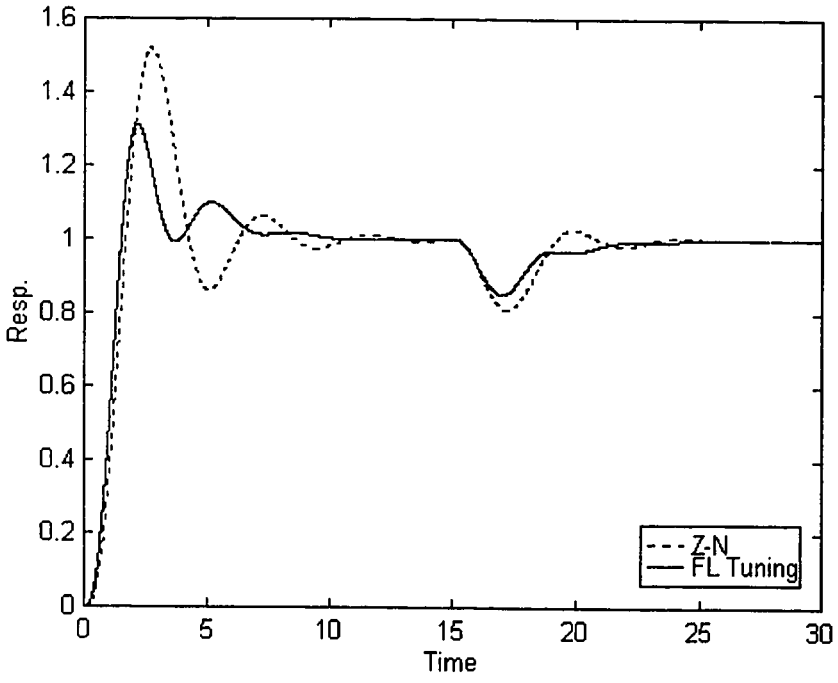


Figure 3.9 : *Simulation results for example 2*

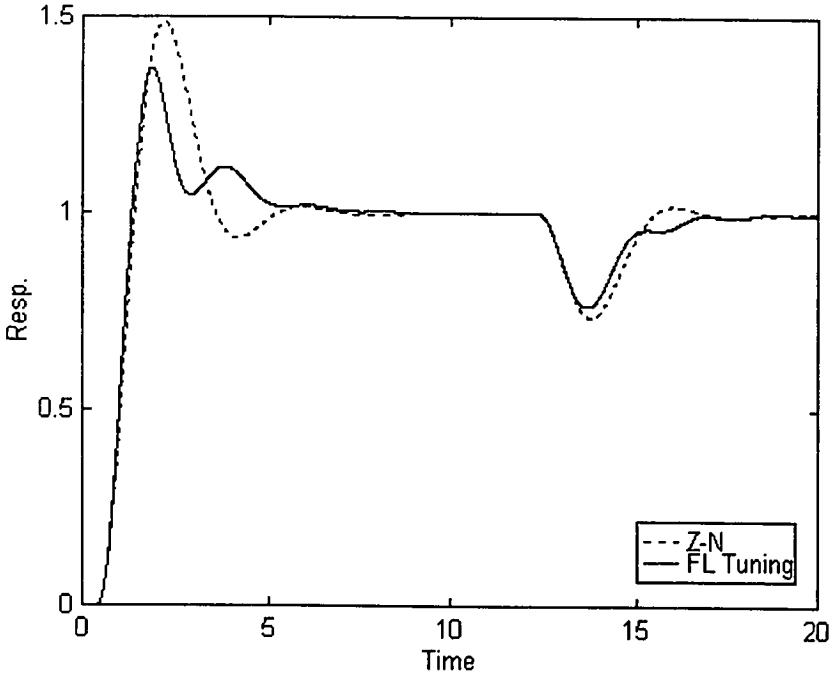


Figure 3.10 : *Simulation results for example 3*

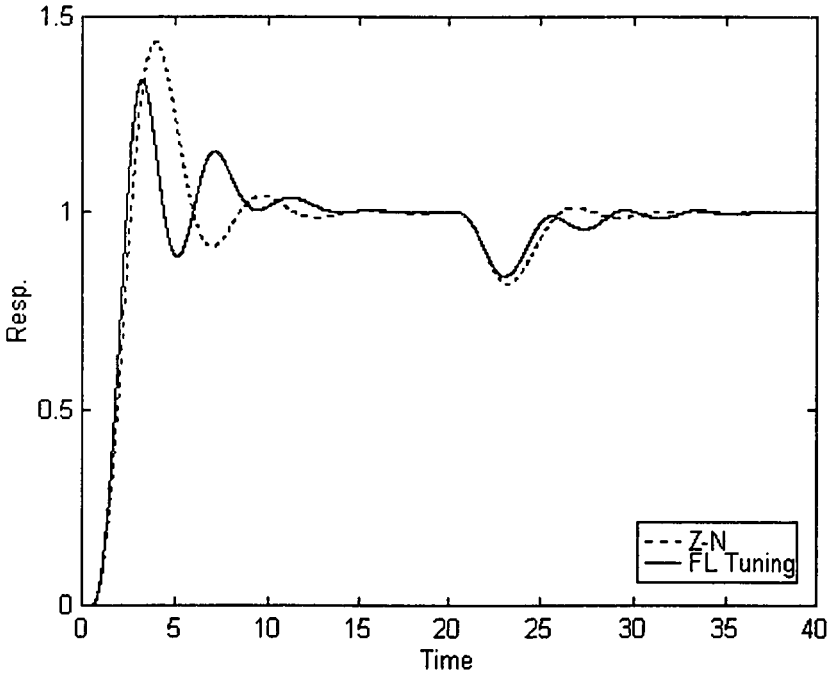


Figure 3.11 : *Simulation results for example 4*

Table 3.4 : *Summery of simulation results*

	Z-N		FL ST	
Example	ISE	IAE	ISE	IAE
1	0.2968	0.6895	0.2279	0.5629
2	1.2812	2.6584	1.0582	2.2068
3	1.1778	2.1697	1.0053	1.9224
4	2.0223	3.6031	1.6722	3.1608

The first and major difference between the proposed scheme in this chapter and the one presented in [15] is that it is clear that using only one variable (i.e., α) to adjust all PID parameters means coupling the proportional gain K_c to the integral gain ($K_i = K_c / T_i$) and derivative gain ($K_d = K_c * T_d$). Now, by de-coupling the tuning of K_c from K_i and K_d, we gain another degree of freedom and, consequently, more flexibility in the tuning of the PID parameters. Such flexibility results in an improved system's performance. In our scheme, we made use of the fact that varying β would improve the performance. Therefore, we have used β as a parameter to adjust the contribution of the proportional term in the PID equation. By doing so, we have decoupled the adaptation of the proportional term from that of the integral and derivative gains.

Our approach is also different from the one in [15] in that we avoided using the recursive equations that update α (i.e., Equations 3.14). When we experimented with these equations, resulted in an oscillatory response for some systems that needed small sampling times.

To illustrate the above mentioned differences, the following fourth order system has been used in different examples.

$$G(s) = \frac{27}{(s+1)(s+3)^3} \qquad (3.15)$$

with the following scaling factors:

sfe = 0.25, *sfde* = 0.25, sfα = 1.3, *sf*β = 0.5

Figure 3.12 depicts the response obtained when using He's scheme [15], and Figure 3.15 shows the improvement obtained when using our scheme, where we have de-coupled the tuning of K_c from K_i and K_d . We can observe from this example that the scheme outlined in [15] failed to perform satisfactorily while the modified proposed scheme provided an acceptable performance. In order to study the effect of coupling the PID parameters, the above process has been simulated using our scheme where the PID parameters have been coupled with the parameter α (see Figure 3.16) . Comparing Figures 3.15 and 3.16, one can clearly see the effect of coupling the PID parameters.

Figure 3.13 shows the effect of reducing the sampling time from ts = 0.2s to ts = 0.05s on the performance of He's scheme. The response become even more oscillatory than in the case where ts = 0.2, and the performance in both cases (i.e., Figure 3.12 and Figure 3.13) is not satisfactory. To explain the effect of α on the location of the poles of the closed loop system, we generated the loci of the roots of the closed loop characteristic equation as a function of α. Figure 3.14 shows the root locus of the closed loop system with the process given by (3.15) and the PID tuned with He's scheme, where the value of α varies between 0 and 2 (the range of α generated by He's scheme is between 0 and 1 only). It is obvious form Figure 3.14 that, at about α = 1, two of the roots cross to the instability region (i.e., the right half plan). This explains the oscillation obtained from He's scheme in Figure 3.13, where we see the value of α was continuously increasing and it was approaching the value 1.

In [35], only the proportional gain K_c and the derivative gain K_d are varied using FL inference whereas the integral gain is determined from the derivative time constant. The algorithm in [35] can not, hence, be applied for a PI controller instead of a PID. In addition, its is assumed in [35] that K_c and K_d are in prescribed ranges [$K_{c,min}$, $K_{c,max}$] and [$K_{d,min}$, $K_{d,max}$], respectively. These ranges are determined empirically according to the following [35] :

$$K_{c,\min} = 0.32K_u \ , \ K_{c,\max} = 0.6K_u$$
$$K_{d,\min} = 0.8K_uT_u \ , \ K_{d,\max} = 0.15K_uT_u \qquad (3.16)$$

One can observe that $K_{c,\,min}$ is taken approximately as half the Z-N value while $K_{c,\,max}$ is equal to the Z-N value. On the other hand, $K_{d,\,min}$ is equal to the corresponding Z-N value and $K_{d,max}$ is equal to twice the corresponding Z-N value. The adaptation scheme, hence, starts from the minimum values of the parameters and, using FL adaptation, the parameters are varied in the prescribed range. The following process has been simulated in [35] :

$$G(s) = \frac{e^{-0.5s}}{(s+1)^2} \qquad (3.17)$$

and the following results, in terms of ISE and IAE, have been reported :

Z-N	Scheme in [35]	% Improvement over Z-N
IAE = 1.37	IAE = 1.18	13.9 %
ISE = 0.871	ISE = 0.772	11.4 %

When simulating the above process using our algorithm, the following results were obtained (see Figure 3.17) :

Z-N	Our scheme	% Improvement over Z-N
IAE = 2.0504	IAE =1.5577	24.1 %
ISE = 1.2841	ISE = 1.1687	9 %

It is observed from the above simulation results that our algorithm provided a big improvement in the IAE compared to the scheme in [35] which was slightly better than our scheme in terms of the improvement in the ISE.

The fuzzy pre-compensated PID controller in [18] is based on the idea of generating a modified reference setpoint. This modified setpoint is simply the sum of the actual setpoint and a value generated from a FL scheme. The inputs to the FL scheme are the error (e) and the difference of the error (Δe). Therefore, all three terms of the PID controller will operate on an error signal that is modified, indirectly using FL inference mechanism.

We noticed that, with the algorithm of [18], the settling time is large and sometimes steady state error exists in some of the step responses generated using the scheme in [18]. This problem is due to the modification of the error signal in all of the parameters of the PID controller. Our scheme overcomes this problem by modifying the proportional part of the error signal only and keeping the actual error signal for both the derivative and integral terms unaltered, and varying their gains instead. Our scheme is also different in the sense that it includes β as an explicit parameterization of the contribution of the error in the proportional part of the PID controller, which makes it easier to derive the FL rules.

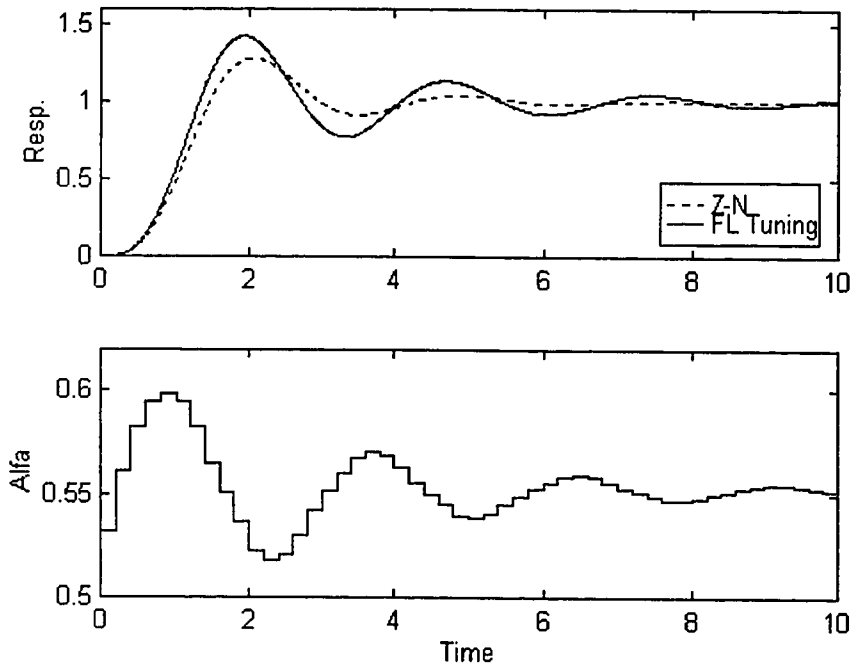


Figure 3.12 : *He's scheme (sampling time =0.2)*

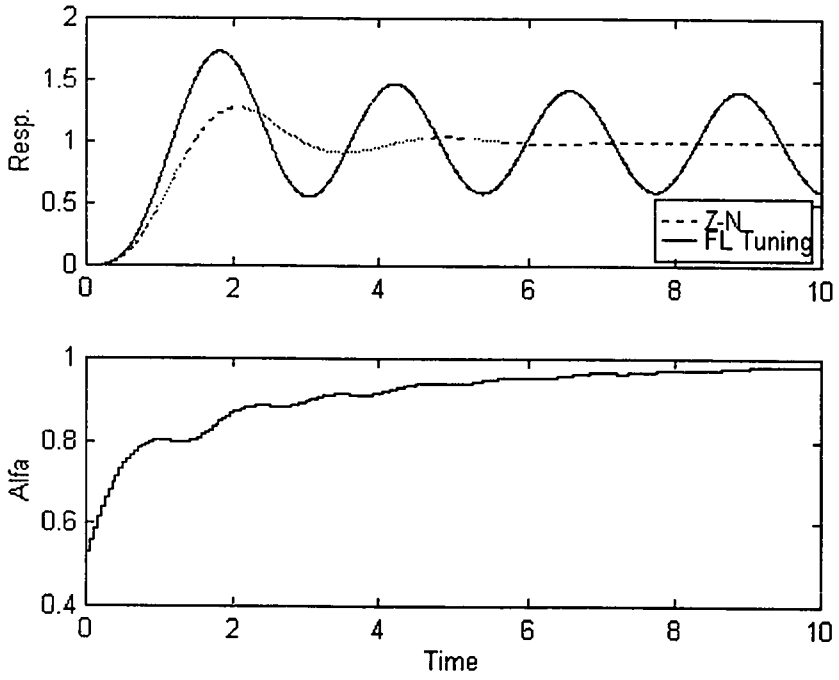


Figure 3.13 : *He's Scheme (sampling time =0.05)*

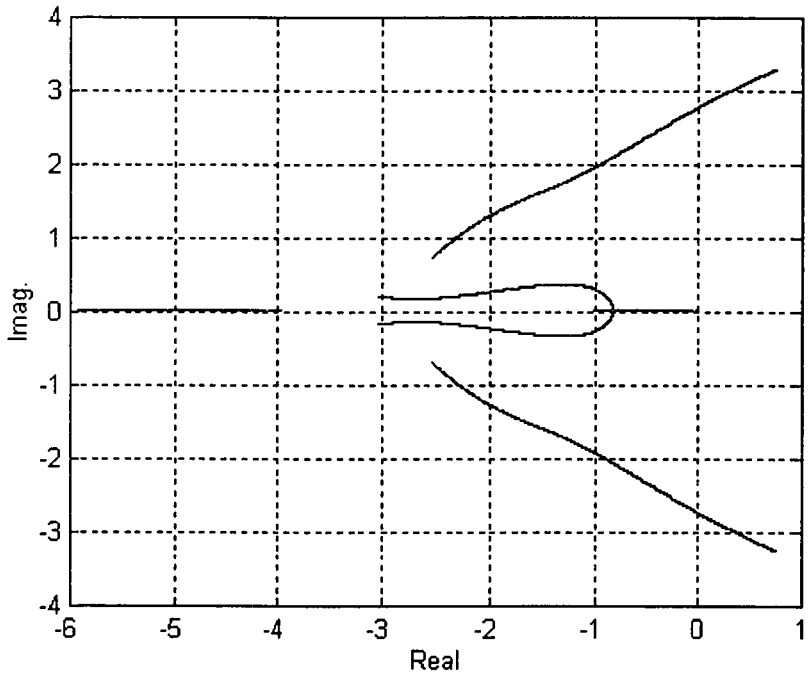


Figure 3.14 : *Root locus of the closed loop system of the process given by (3.15) and the PID tuned using He's scheme*

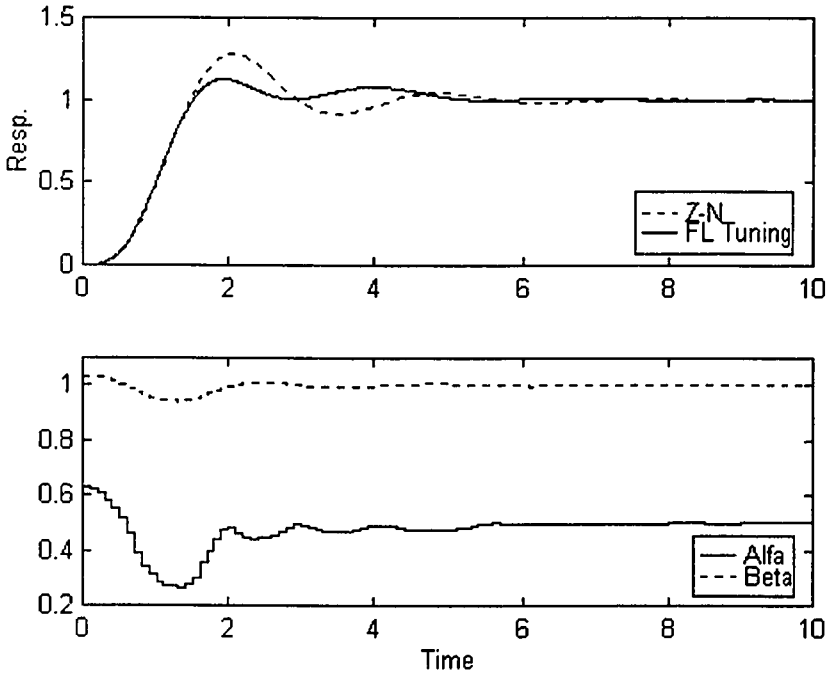


Figure 3.15 : *Our scheme (sampling time =0.2)*

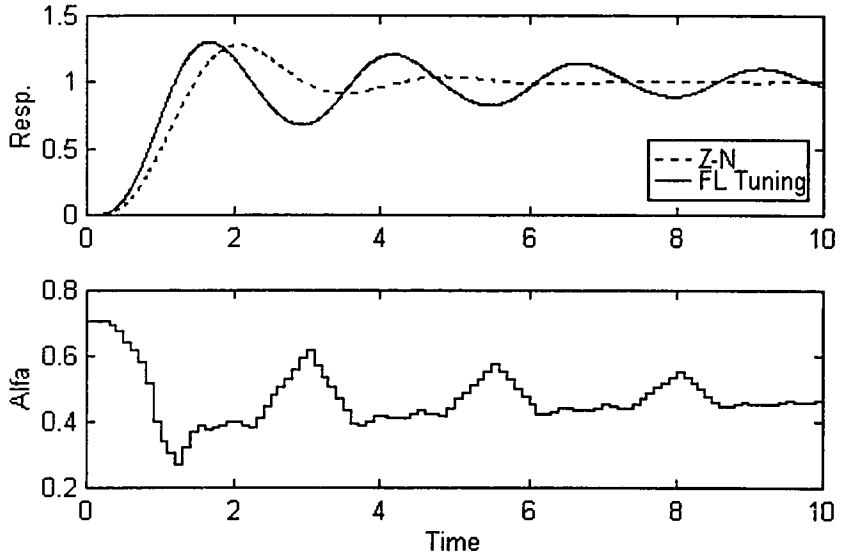


Figure 3.16 : *Effect of coupling the PID parameters on systems performance*

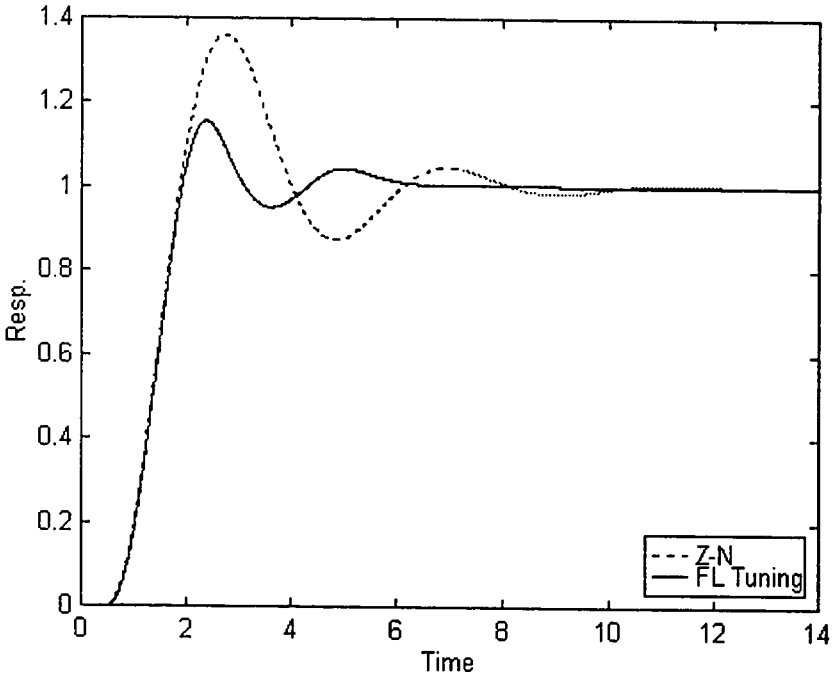


Figure 3.17

3.9 Conclusions

PID controllers are widely used in industry and, most likely, will continue to be so. Any attempt at improving their performance and capabilities using advanced control techniques would be very beneficial. In this chapter of the thesis, we utilized the power of fuzzy logic control to enhance the performance of the PID controller and to capture the control engineer's intuition and experience into fuzzy control rules that would take over the tuning function and, consequently, deliver a superior performance over the PID controller tuned using the Ziegler-Nichols method. The simulations results showed that the proposed scheme can produce marked improvements in the performance of the PID controller. The computational power required to implement the proposed control scheme is very low. In addition, a small look-up table of fuzzy values calculated off-line can be used to determine the proper tuning parameters. The memory requirements for this look-up table is quite low and it is available in today's modern control systems.

CHAPTER 4

FL-BASED TIME DELAY COMPENSATION

In this chapter, a FL-based time delay compensation controller is presented. The controller is an extension of the algorithm outlined in chapter 3.

4.1 Introduction

As have been mentioned in the Chapter 1, a time delay in a feedback loop is often a serious obstacle to obtaining an acceptable performance. Such a delay prevents high controller gains from being used, leading to poor and sluggish system response when using conventional control methods such as PID control.

It has been known from practical experience that the dead time to time constant ratio of a process (defined by Ho [16] as the normalized dead time) may be used as a measure of the difficulty of controlling the process. A process with small normalized dead time (smaller than 2) is

easy to control while one with a large normalized dead time is difficult to control [16]. The normalized dead time can be calculated using the open loop Z-N technique presented in section 2.3.1. Hence, a process with a dead time to time constant ratio (i.e., normalized dead time) greater than 2 is considered to be a process with large dead time.

The above discussion indicates that to control processes with a large dead time effectively, a more advanced control technique such as model-based predictive control is required. A well known model based, dead time compensating controller is the Smith Predictor (SP). The basic structure of the SP is shown in Figure 4.1. It requires a model, G_{pm} , of the process, G_p , to predict future process outputs. A controller G_c is designed to shape the response. The derivative part of the PID controller is not usually used with the SP since the prediction is already performed by the dead time compensation scheme [11].

The basic concept behind the SP is the use of a model of the process free from time delay to predict the effect of current control actions on the future outputs. This effectively removes the dead time from the dynamics of the loop and, hence, making it possible to operate the loop at higher gains than is otherwise possible. With a SP, the design problem is converted to a design problem for the classical time delay-free system.

The disadvantage of the SP is that a model of the process to be controlled needs to be developed. In addition, the performance of the SP is quite sensitive to the accuracy of the model parameters, especially the time

delay value. Therefore, if the process dynamics or the time delay change significantly, the model will be inaccurate and the performance of the algorithm will deteriorate and probably causing instability in the closed loop [28].

To overcome the above mentioned problems of the SP sensitivity to modeling errors and dead time variations, one has to resort to some kind of adaptive control techniques. It is believed that the proposed FL-based ST scheme presented in chapter 3 might be a suitable alternative to conventional adaptive control methods in these situations, and can be quite useful when incorporated in the SP. We envision that the FL-based ST scheme combined with SP can accommodate some of the model uncertainties that might take place when using SP.

Therefore, in this chapter, we propose to use the FL-based ST scheme, developed in chapter 3, to handle processes with large dead times. The scheme is built upon the basic simple adaptive relay feedback auto-tuner [7] which is quite easy to design and implement. The scheme is used in conjunction with SP where the FL-based self-tuning algorithm will continuously monitor the process and adjust the PI controller gains in such a way so that an acceptable performance is obtained.

The chapter is organized as follows. A brief review of the SP method is given in section 4.2 and the modeling technique applied in the proposed algorithm is outlined in section 4.3. We present the proposed FL-based SP self-tuning scheme in section 4.4, followed by simulations of different processes in section 4.5, and conclusions in section 4.6.

4.2 Time delay compensation via SP

Figure 4.1 shows the basic configuration of the SP, where the process model has been divided into two parts : the time delay-free part G_m and the pure time delay part e^{-sL_m}. The controller (i.e., PI) acts on the predicted output $y(t+L)$, where L is the time delay. This is accomplished by letting the controller operates on the simulated process output y_1 which is time delay free. Assume that the process is described by :

$$G_p(s) = G(s)\, e^{-sL} \tag{4.1}$$

Where $G(s)$ is a rational transfer function. The model of the process is given by:

$$G_{pm}(s) = G_m(s)\, e^{-sL_m} \tag{4.2}$$

Figure 4.2 shows an equivalent block diagram to Figure 4.1. The closed loop transfer function between the setpoint y_r and the process output y becomes :

$$\frac{y}{y_r} = \frac{G_c G e^{-sL}}{1 + G_c G_m - G_c G_m e^{-sL_m} + G_c G e^{-sL}} \tag{4.3}$$

In case of perfect modeling, (i.e., $G = G_m$ and $L = L_m$), equation (4.3) reduces to :

$$\frac{y}{y_r} = \frac{G_c G}{1 + G_c G_m} e^{-sL} \tag{4.4}$$

Equation (4.4) indicates that the characteristic equation is free of the time delay and that the effect of the dead time is moved outside the loop. This means that the design problem is one of designing the controller G_c for the time delay-free process G. Figure 4.3 shows the block diagram representation of Equation (4.4) which is exactly equivalent to Figure 4.1 in the case of perfect matching between the process and model. It is clear from Figure 4.3 that the dead time effect has been moved outside the loop. This property of completely moving the dead time outside the loop will be lost if the model of the process is severely inaccurate. A Number of researchers (see [10,28] for example) have shown, however, that the SP can still provide improvement over conventional feedback if the model parameters are within ± 30% of the real ones.

4.3 Modeling mechanism

The most important step in implementing the SP is to identify a model of the process dynamics as accurately as possible. There are various techniques to identify process models such as the recursive identification techniques used in adaptive control [2]. These methods are quite complex and require a considerable computational power. In addition, a certain model structure needs to be assumed. A much simpler identification (non-parameteric) technique, that have been used in this chapter and in the scheme we have developed in chapter 3, is the relay feedback technique [7] in which useful information about the process is

extracted from the resulting controlled limit cycle oscillations. This technique is simple and easy to implement and, in addition, only limited prior knowledge about the process is needed.

Most stable industrial processes can be described by a rational transfer function plus dead time :

$$G_m(s) = \frac{K_p\, e^{-sL_m}}{(1+sT_m)^n} \tag{4.5}$$

where n = 1 or 2, is the order of the model [13]. The model order is determined based on prior knowledge of the process. In the absence of that knowledge, it is recommended that an order of two be used [13]. Once K_u and T_u are determined from the relay experiment, one can calculate the model parameters (i.e., the values of K_P , L_m ,and T_m). The process gain is determined after the first setpoint change as follows:

$$K_p = \Delta y / \Delta u \tag{4.6}$$

Where Δy and Δu are the process output and input changes respectively. If a first order model is chosen, then the dead time L_m and time constant T_m are calculated from the following equations (see [13] for more details) :

$$T_m = \frac{T_u}{2\pi}\sqrt{\left(K_u^2 K_p^2 - 1\right)} \tag{4.7}$$

$$L_m = \frac{T_u}{2\pi}\left[\pi - 2\tan^{-1}\left(\frac{2\pi T_m}{T_u}\right)\right] \tag{4.8}$$

Similarly, if a second order model is chosen (i.e., n = 2), then the dead time L_m and time constant T_m are calculated from the following equations [13] :

$$T_m = \frac{T_u}{2\pi}\sqrt{\left(K_u K_p - 1\right)} \tag{4.9}$$

$$L_m = \frac{T_u}{2\pi}\left[\pi - 2\tan^{-1}\left(\frac{2\pi T_m}{T_u}\right)\right] \tag{4.10}$$

Once the model parameters are determined using the above equations, the critical point (i.e., the point at which the ultimate gain K_u and ultimate period T_u occur) of the time delay free system can be obtained using the relay feedback experiment. Consequently, using Ziegler-Nichols equations, we obtain the initial PI tuning parameters, which will be varied using FL as explained in the next section.

4.4 Proposed Controller

Figure 4.4 shows a general configuration of the proposed algorithm, where we can see that the FL-based ST scheme is added to the classical SP. The FL tuning scheme, based on the error *e(k)* and change of error *Δe(k)*, generates appropriate corrections to the controller's gains which are parameterized by the two parameters α and β . The controller is straightforward extension of the FL-based ST scheme presented in chapter 3.

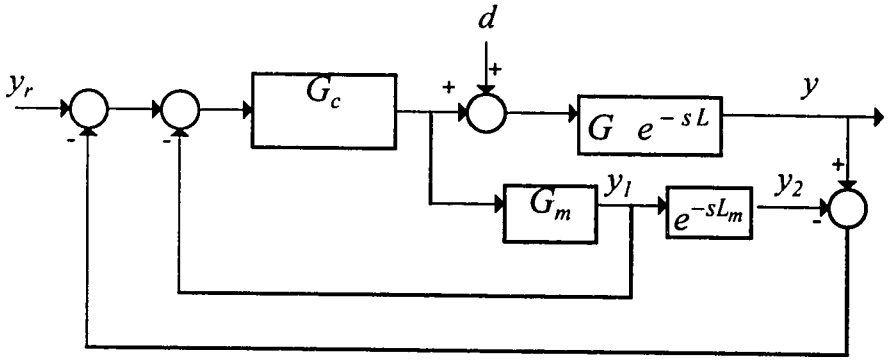


Figure 4.1 : *Smith Predictor Control*

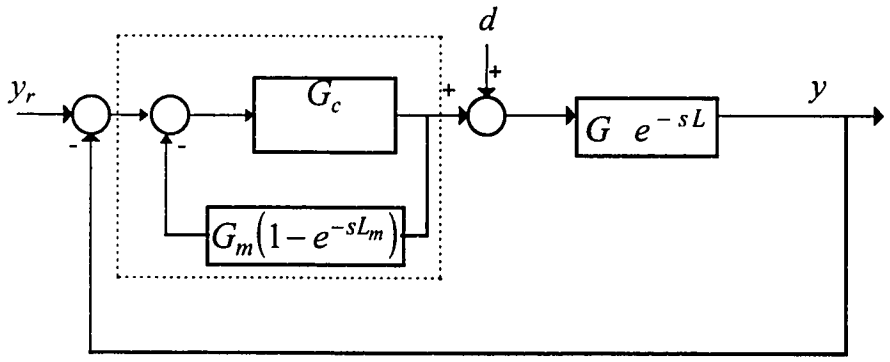


Figure 4.2 : *Alternative SP Configuration*

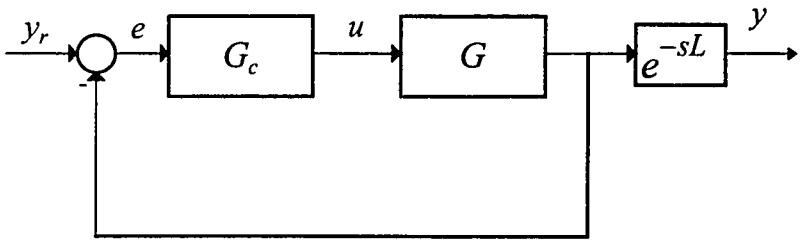


Figure 4.3 : *Reduced SP Block Diagram*

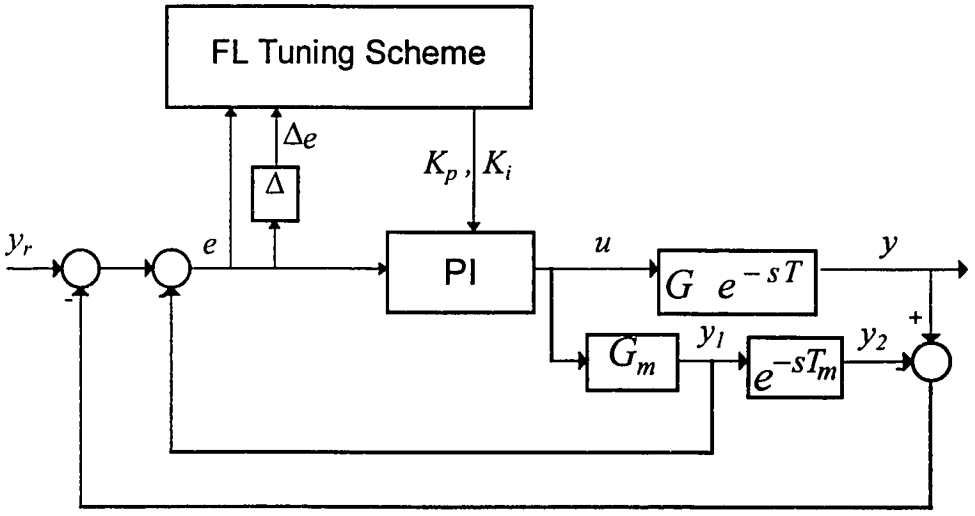


Figure 4.4 : *SP FL-Based Controller*

The proposed PI controller has the following model:

$$u(t) = K\left[\left(\beta y_r - y\right) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau\right] \qquad (4.11)$$

In this scheme, the FL is used to tune the PI controller while the SP compensates for the long dead time that is commonly associated with different industrial processes. The key idea behind the ST scheme is the same as the one already covered in chapter 3 except for the fact that, here, a PI controller is used instead of a PID. Therefore, only one of the Z-N equations is parameterized with the parameter α so that the integral time constant, T_i , is varied according to :

$$T_i = \quad 1.275\frac{1}{1+\alpha}T_u \qquad (4.12)$$

and the proportional contribution is varied through the variation of the parameter β. Again, here, for α = 0.5, equation (4.12) reduces to the Z-N equation.

4.5 Simulations

Simulations of several plant models have been conducted to study the new algorithm. The setpoint response of the new algorithm for each of

the models simulated have been compared with the responses obtained from the original setup of the SP.

Example 1

The closed loop response and the trends of α and β for the following process are shown in Figures 4.5 and 4.6.

$$G_1 = \frac{e^{-4s}}{(s+1)(0.5s+1)} \tag{4.13}$$

Figure 4.5 shows the response for an exact matching , i.e., $G_p = G_{pm}$. Figure 4.6 shows the results when using the following model in the SP.

$$G_m = \frac{e^{-4s}}{(0.85s+1)(0.4s+1)} \tag{4.14}$$

We can see, from Figures 4.5 and 4.6, the marked improvement in the responses obtained with our FL scheme, especially in terms of overshoot and settling time. The rise time obtained from our scheme was just about the same as the one from the original SP. It is clear from Figure 4.6 that the FL-based ST scheme has reduced the effect of mismatch between the process and the model on the system's performance and made the SP less sensitive to the model errors in this particular example.

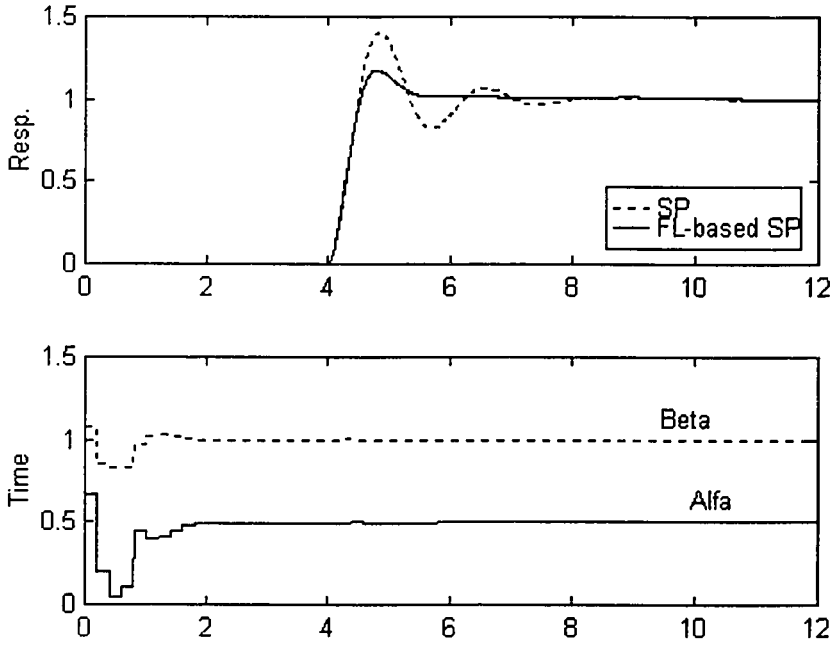


Figure 4.5 : *Simulation results for example 1 (Perfect modeling)*

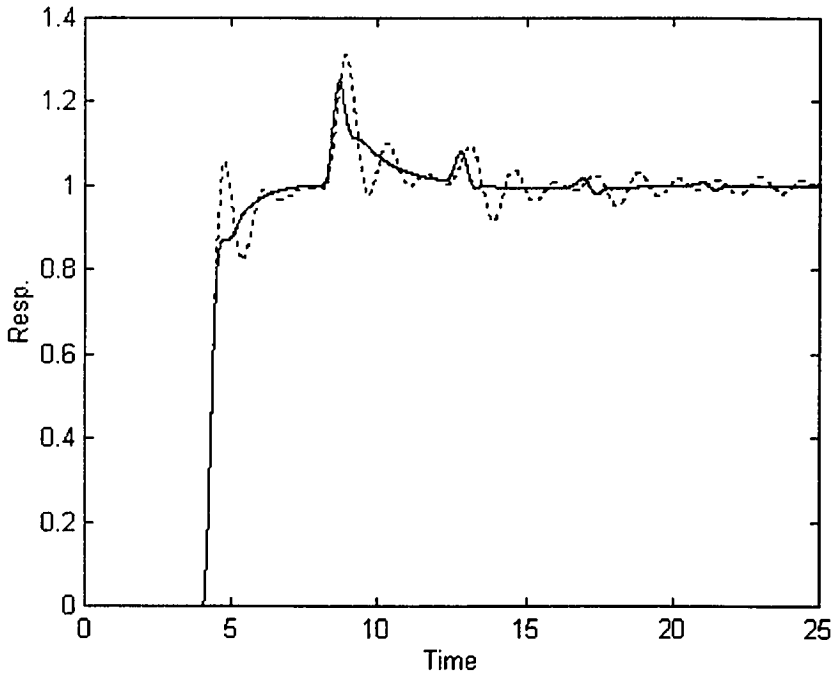


Figure 4.6 : *Simulation results for example 1 (perturbed model)*

Example 2

The following system :

$$G_2 = \frac{0.57e^{-18.7s}}{(8.6s+1)^2} \tag{4.15}$$

is approximated by the following second order model, using the identification technique presented in section 4.3 :

$$G_m = \frac{0.57e^{-18.72s}}{(8.49s+1)^2} \tag{4.16}$$

The results of the simulation shown in Figure 4.7 demonstrate the improvement obtained with the proposed algorithm. The response of the FL-based scheme was superior to the SP's response, especially, in terms of reducing the overshoot, undershoot and settling time. The tuning parameters used in this simulation example are as follows :

sfe = 1, $sfde$ = 0.2, $sf\alpha$ = 0.67, $sf\beta$ = 0.5, T_u = 18.78, K_u= 20.08

Example 3

The fourth order system

$$G_3 = \frac{e^{-10s}}{(s+1)(0.5s+1)(0.25s+1)(0.125s+1)} \tag{4.17}$$

is modeled by the following second-order model, using the equations (4.9)and (4.10) :

$$G_m = \frac{e^{-10.29s}}{(0.75s+1)^2} \tag{4.18}$$

The tuning parameters used in this simulation example are as follows :

sfe = 1, $sfde$ = 0.5, $sf\alpha$ = 1, $sf\beta$ = 0.67, T_u = 1.77, K_u= 10.15

The responses for both the SP and the FL scheme are shown in Figure 4.8, where we can see again the improvement gained by the new scheme in terms of settling time and overshoot reduction. Once more, we see the marked recovery of the system's response using the FL-based self-tuning scheme in the face of the model uncertainties.

Example 4

This last example treats the following third order process :

$$G_4 = \frac{e^{-4s}}{(s+1)^3} \tag{4.19}$$

which is modeled by equations (4.9)and (4.10):

$$G_m = \frac{e^{-4.52s}}{(1.25s+1)^2} \tag{4.20}$$

Using the following tuning parameters, we get the simulation results shown in Figure 4.9 :

sfe = 1, $sfde$ = 1, $sf\alpha$ = 1, $sf\beta$ = 0.67, T_u = 2.25, K_u= 15.41

The simulation results demonstrate the capability of the new scheme in accommodating uncertainty in the process model. When the SP almost fails to cope with the model-process mismatch, the FL algorithm provides acceptable results shown in Figure 4.9. We notice that the response from the new scheme settled at the desired setpoint with minimum oscillations compared to the original SP where oscillations persist for quite long time.

The results of the above simulation examples indicate, in general, that the new scheme reduced the system's sensitivity to modeling errors. In addition, both the overshoot and settling time have been considerably reduced in all examples. The rise times obtained with the FL scheme and the original setup of the SP were about the same.

The ISE and the IAE performance indices have been calculated, as outlined in section 2.3.3, for the above simulation examples and presented in Table 4.1. The content of Table 4.1 shows the improvement in performance in terms of the ISE and IAE obtained from the new scheme in all the examples considered.

4.6 Conclusions

A FL-based auto-tuning scheme for the smith predictor has been presented in this chapter. The motivation was to overcome the deficiency of classical SP controllers in accommodate model uncertainties. The simulation results have shown that the combination of SP and the FL-based ST algorithm can provide improved performance over the original SP and that the sensitivity of the SP to model inaccuracies can be reduced considerably. Furthermore, the new algorithm can give acceptable time response in situations where the classical setup of the SP might fail practically. The design of the FL-based SP scheme is quite simple and easy to implement. In the design procedure, we make good use of the relay feedback experiment to derive the model of the process and to obtain the initial PI controller parameters as well.

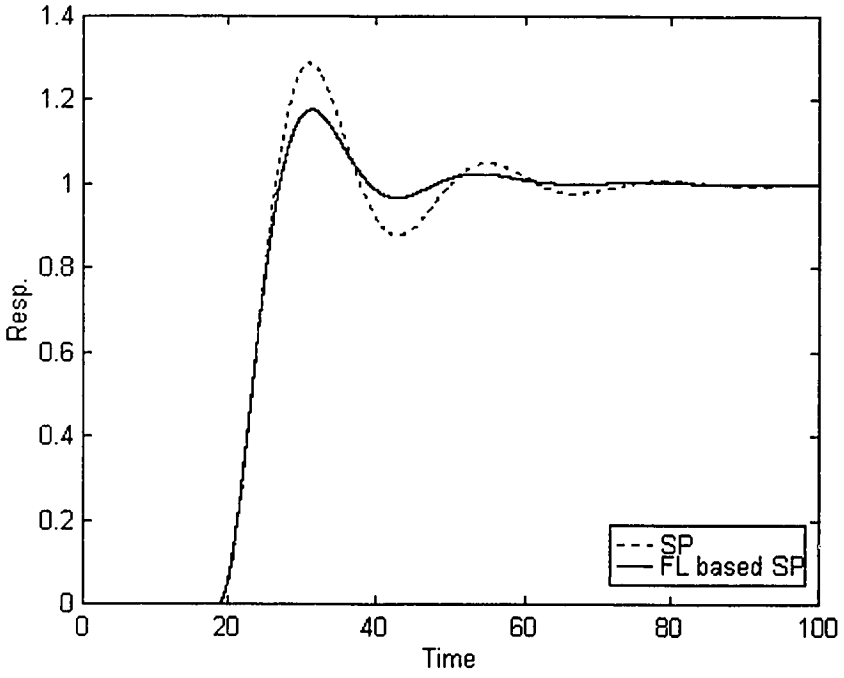


Figure 4.7 : *Simulation results for example 2*

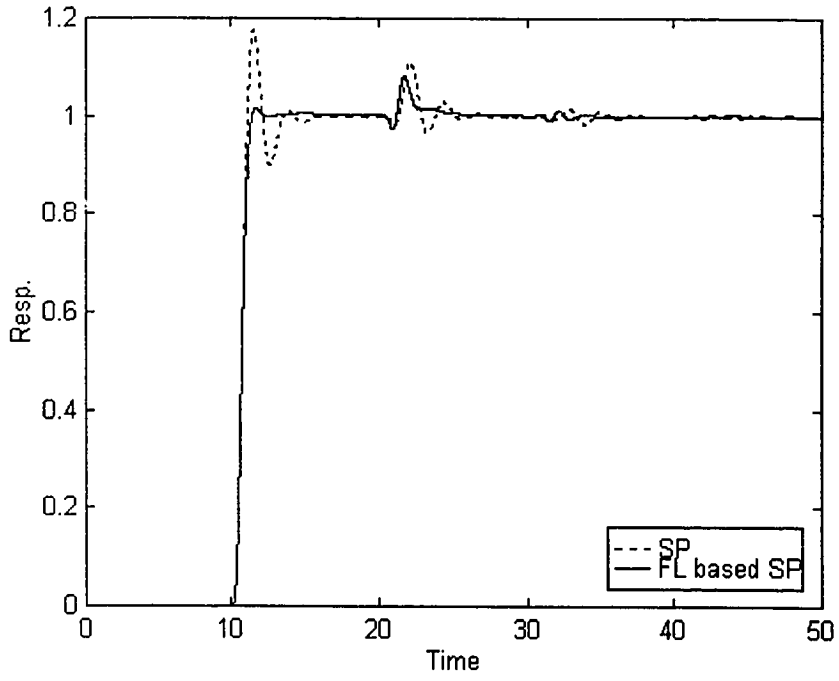


Figure 4.8 : *Simulation results for example 3*

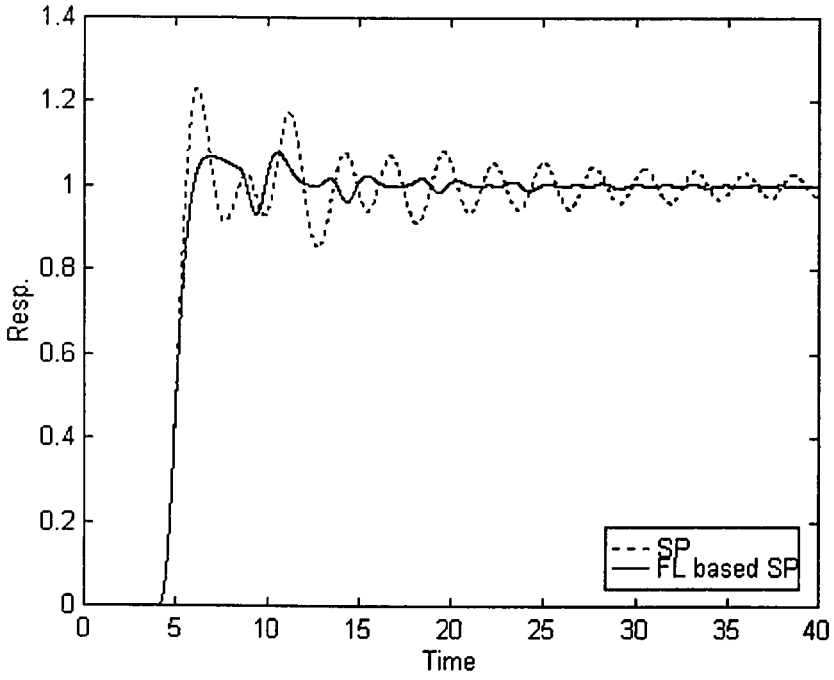


Figure 4.9 : *Simulation results for example 4*

Table 4.1 : *Simulation results summery*

	SP		FL SP	
	ISE	IAE	ISE	IAE
G_1 (perfect Modeling)	4.3158	4.6785	4.2402	4.4569
G_1 (perturbed model)	4.3904	4.7041	4.2433	4.4718
G_2	22.6378	26.7611	22.2178	24.9460
G_3	10.6245	11.1479	10.6002	10.9206
G_4	4.9108	6.5308	4.8207	5.4558

CHAPTER 5

FL-BASED GAIN SCHEDULING CONTROLLER

5.1 Introduction

Many of the processes encountered in practice have nonlinear characteristics. The use of simple controllers, such as PIDs, is often adequate when the process nonlinearities are mild and the operation of the process is restricted to a small region around a nominal steady state. Advanced control methods, such as adaptive control or model-based control, can be used when high performance is required over a broader range of operating conditions. However, these techniques are quite involved and require an accurate model of the process as is the case in model-based control. A simpler alternative, with some loss in performance though, is the gain scheduling control method, where linear controllers are used at different operating points with gain scheduling used at a higher level [22].

Gain Scheduling (GS) is a special case of adaptive control. It is a special type of nonlinear feedback; it has a linear regulator whose parameters are changed as a function of the operating conditions in a preprogrammed way [2]. It is a very useful technique to compensate for variations in the process parameters that can be predicted from measured operating conditions or *known nonlinearities* in the process.

The objective of this chapter is to improve the transition between the different linear controllers by investigating the Fuzzy Gain Scheduling (FGS) method presented by Ling et. al. [22]. The other objective is to refine the performance of the GS controller by using the FL-based ST scheme, presented in chapter 3, at the local controller's level.

This chapter is organized as follows. The conventional GS controller and the FGS controller are briefly covered in sections 5.2 and 5.3, respectively. The FL-based GS algorithm is outlined in section 5.4. Simulation examples are presented in section 5.5 and the conclusion is given in 5.6.

5.2 Gain Scheduling Control

Figure 5.1 shows a block diagram of a conventional GS control system. GS can be viewed as a feedback control system in which the feedback gains are adjusted using feedforward compensation [2]. In general, GS is limited to applications where the process dynamics depend on known,

measurable variables and the necessary controller adjustments are not too complicated [27].

The design of GS control system starts by studying the physics of the process and, based on that, suitable scheduling variables that correlate well with the changes in process dynamics are determined. These variables must reflect the operating conditions of the plant [2]. The next step in the design is to calculate the controller's parameters at specified operating conditions. Special consideration, when tuning the controller, must be given to the transition between different operating points. The major problem with GS is that it is an open loop compensation technique. There is no feedback to correct for inaccurate schedule. This is the reason why many researchers refrain from considering GS as an adaptive control strategy [2].

5.3 Fuzzy Gain Scheduling Control

A FGS controller, as presented by Ling et. al. [22], is basically the same as a conventional GS controller. The process output range is partitioned into several regions, which would overlap if FGS is used. A linear controller is designed for each region. A higher level rule-based FL controller activates the appropriate linear controller by examining the process condition and using fuzzy inference [22].

For a PID controller that has the model given by (2.1) and with two sets of GS parameters, we have the following conventional GS rule:

$$\begin{aligned} &\text{IF } y > y^* \text{ THEN } K_c = K_{c1}\ ,\ T_i = T_{i1}\ ,\ T_d = T_{d1} \\ &\qquad \text{ELSE } K_c = K_{c2}\ ,\ T_i = T_{i2}\ ,\ T_d = T_{d2} \end{aligned} \tag{5.1}$$

Now, if fuzzy logic rules are used with the PID control parameters as the functional consequence, one obtains the following fuzzy rules:

$$\begin{aligned} &\text{IF } y = Y_1 \text{ THEN } K_c = K_{c1}\ ,\ T_i = T_{i1}\ ,\ T_d = T_{d1} \\ &\text{IF } y = Y_2 \text{ THEN } K_c = K_{c2}\ ,\ T_i = T_{i2}\ ,\ T_d = T_{d2} \end{aligned} \tag{5.2}$$

where Y_1 and Y_2 are fuzzy linguistic variables. Figure 5.2 shows an example of the membership functions used in this chapter. Equation (5.2) can be expressed algebraically as :

$$\begin{aligned} K_c &= w_1 K_{c1} + w_2 K_{c2} \\ K_i &= \frac{K_c}{T_i} = \left(\frac{w_1 K_{c1}}{T_{i1}} + \frac{w_2 K_{c2}}{T_{i2}} \right) \\ K_d &= K_c T_d = \left(w_1 K_{c1} T_{d1} + w_2 K_{c2} T_{d2} \right) \end{aligned} \tag{5.3}$$

where the weighting factors w_1 and w_2 are the degree of fulfillment of the antecedents μ_{Y1} and μ_{Y2} of the respective rules at the current operating conditions. In a similar way, the controller's parameters can be expressed in terms of the critical point (i.e., K_u , T_u) as :

$$\begin{aligned} K_{\mathrm{u}} &= w_1 K_{u1} + w_2 K_{u2} \\ T_{\mathrm{u}} &= w_1 T_{u1} + w_2 T_{u2} \end{aligned} \tag{5.4}$$

A common practice in selecting the controller gains is to maintain the closed loop gain K_cK_P at a constant value [22]. Now, assuming that all other elements in the loop have constant gains, the controller gain should be inversely proportional to the process gain. Figure 5.3 shows the controller gain K_c that corresponds to the process gain K_P shown in Figure 5.5. As it is clear from Figure 5.3, FGS provides linear interpolation between the controller gains. This always would be the case when using triangular or trapezoidal fuzzy membership functions.

Comparing the GS and FGS interpolations in Figure 5.3, one can observe that the FGS provides better estimate than the GS, and this estimate becomes even better if the nonlinearity of the process gain is mild. If the nonlinearity of the process gain is severe, however, the FGS provide an estimate which might be much higher than the desired interpolation. This could destabilize the system. A solution to this problem would be to add more fuzzy control rules to the rule base.

The above discussion indicates that while GS changes controller parameters abruptly, FGS interpolates between controller parameters which results in smooth transitions of controller actions between operating conditions.

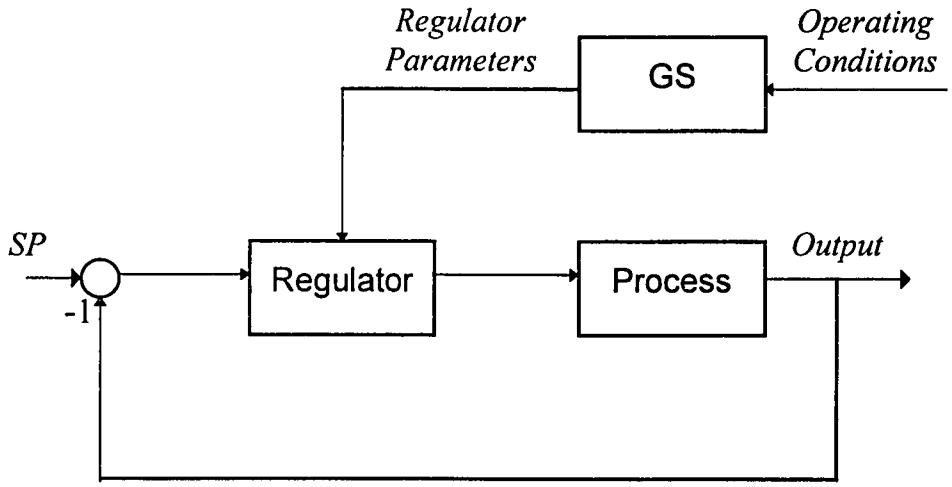


Figure 5.1 : *Gain Scheduling Control*

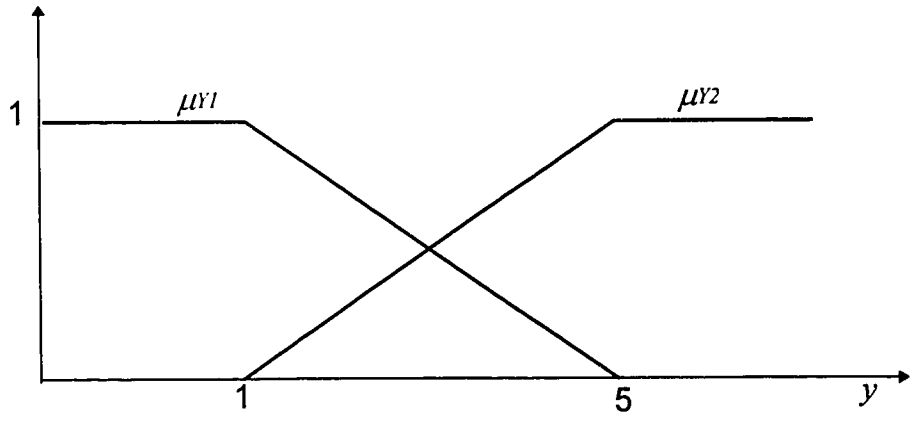


Figure 5.2 : *Membership Functions*

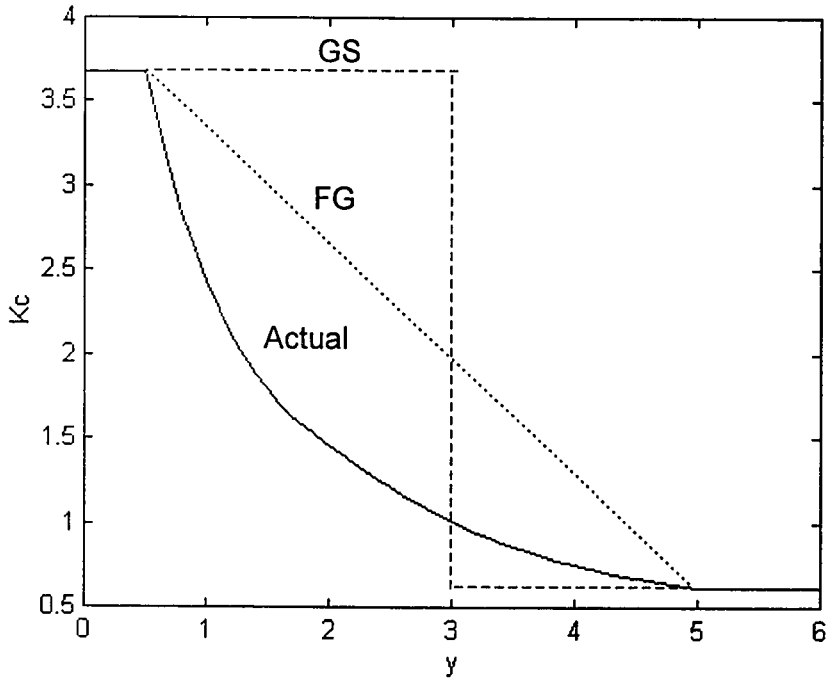


Figure 5.3 : *GS vs. FGS*

5.4 FL-based GS Controller

Figure 5.4 shows the general configuration of the proposed FL-based GS algorithm. The FGS part, as outlined in section 5.3, determines, based on the process output, the operating point in terms of K_u and T_u . The operating point is passed on to the FL tuning scheme which, in turn, determines the controller parameters based on the operating point, the error signals and the resulting FL rules. The details of the FL tuning scheme are covered in chapter 3.

The membership functions used in the FGS are problem-specific. The number of the membership functions depends on the number of controllers used in the GS scheme and, consequently, on the number of rules employed by the FGS. In addition, the shape and the size of the membership functions will be determined based on the process characteristics (e.g., characteristics of the nonlinearity), and on the type of interpolation required between the local controllers. Either triangular or trapezoidal functions should be used if linear interpolation is required.

In addition to interpolating between the local controllers, the fuzzy membership functions are used to determine the best scaling factors of the FL tuning scheme. This will be done by interpolating between scaling factors designed for the individual controllers. For instance, if two GS sets are used then the scaling factors are calculated as :

$$
\begin{aligned}
sfe &= w_1 \cdot sfe_1 + w_2 \cdot sfe_2 \\
sfde &= w_1 \cdot sfde_1 + w_2 \cdot sfde_2 \\
sf\alpha &= w_1 \cdot sf\alpha_1 + w_2 \cdot sf\alpha_2 \\
sf\beta &= w_1 \cdot sf\beta_1 + w_2 \cdot sf\beta_2
\end{aligned}
\tag{5.5}
$$

where sfe , $sfde$, $sf\alpha$, $sf\beta$ are the scaling factors for the error, first difference of the error, α and β , respectively.

The design procedure for the FL-based GS algorithm is as follows:

1. Design the PID controllers at the specified operating points using the relay auto-tuner.
2. Obtain the proper scaling factors for the FL-based ST algorithm at these points.
3. Determine the membership functions that will be used in the FGS and in determining the suitable weighting factors.

5.5 Simulations

The simulations conducted here are in two parts. In the first part, the performance of the FGS is compared to that of the traditional GS. In the second part, the FL-based GS scheme is tested against the FGS through several simulation examples.

Example 1-1

The closed loop responses for the following third order process with GS and FGS are shown in Figures 5.5 and 5.6 :

$$G_1(s) = \frac{K(y)}{(s+1)^3} \tag{5.6}$$

The process gain varies according to the following (see Figure 5.5 (a)) :

$$K(y) = \begin{cases} 2 & \text{if } y \le 0.5 \\ 2y+1 & \text{if } 0.5 < y \le 5 \\ 11 & \text{if } y > 5 \end{cases} \tag{5.7}$$

The simulation results demonstrate the improvement in the closed-loop response obtained with the FGS. It is clear from Figure 5.5 that the response of the GS becomes very oscillatory with the second set point change. It is obvious from Figure 5.6, which shows the trajectory of the proportional gains for both the GS and FGS schemes, that the GS changes the controller parameters abruptly which caused the oscillations. The FGS, however, interpolates between the controller parameters which resulted in smooth transitions of controller actions between the various regions.

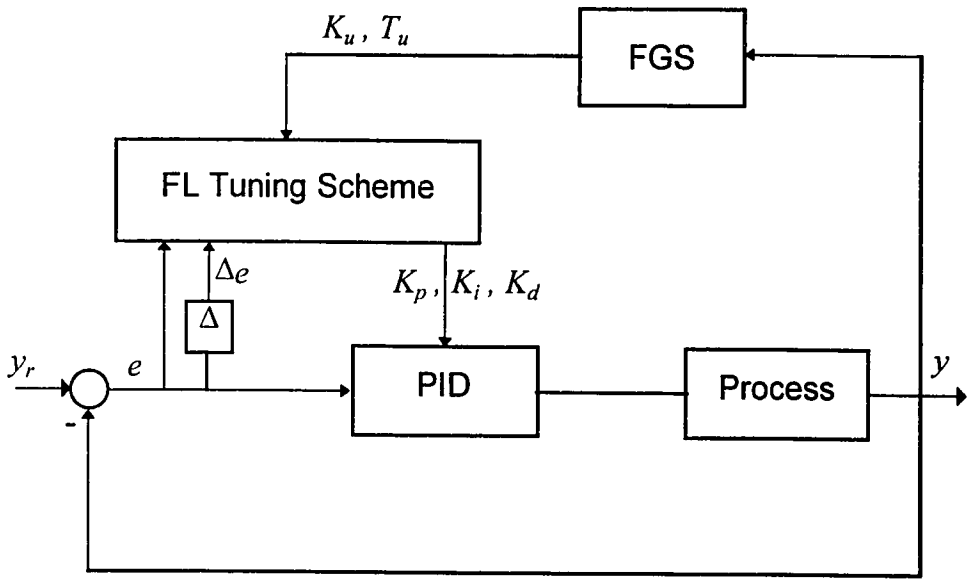


Figure 5.4 : *FL-based GS Controller*

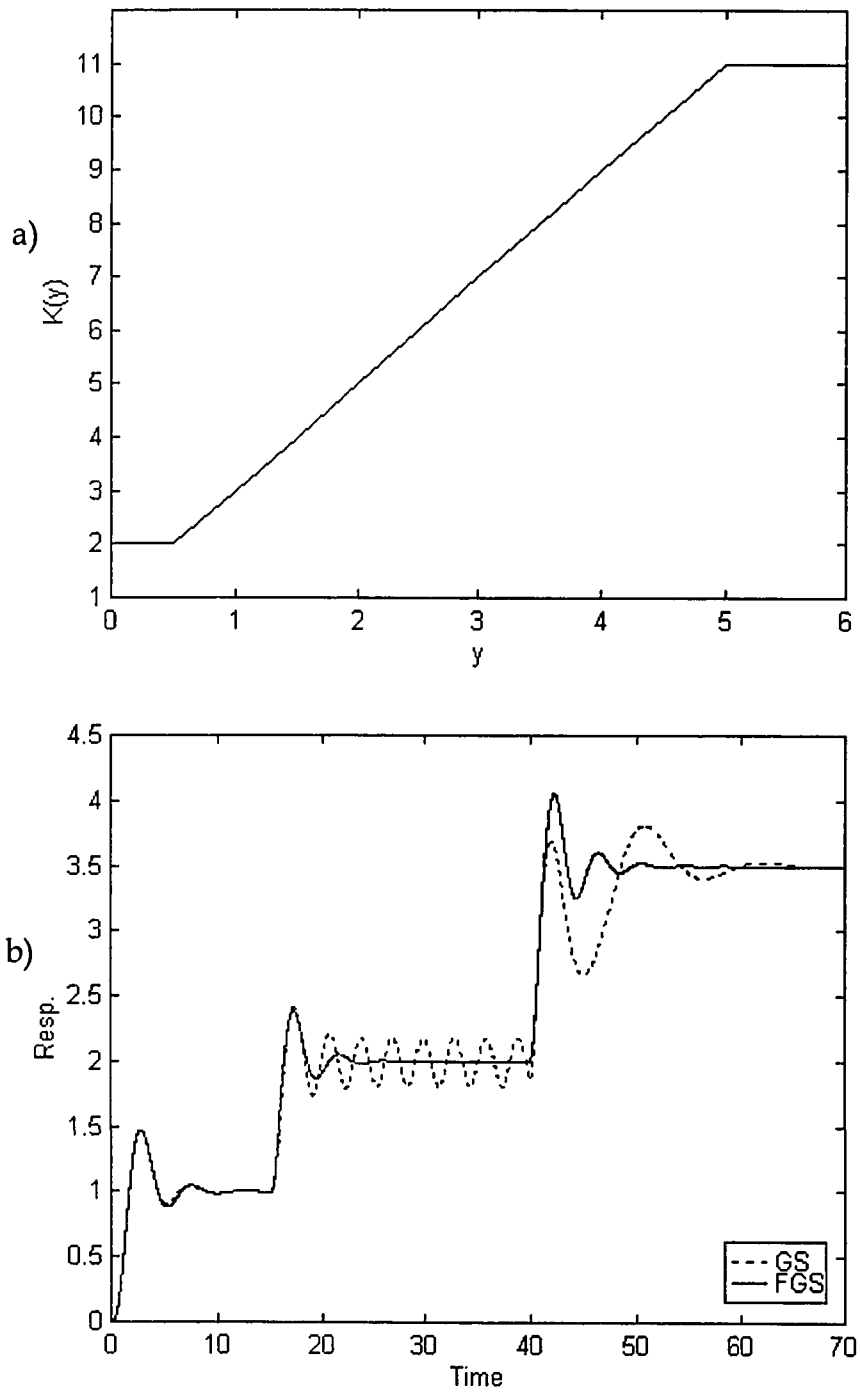


Figure 5.5 : *Simulation Results for example 1-1 , a) Process gain , b) Process outputs with GS and FGS*

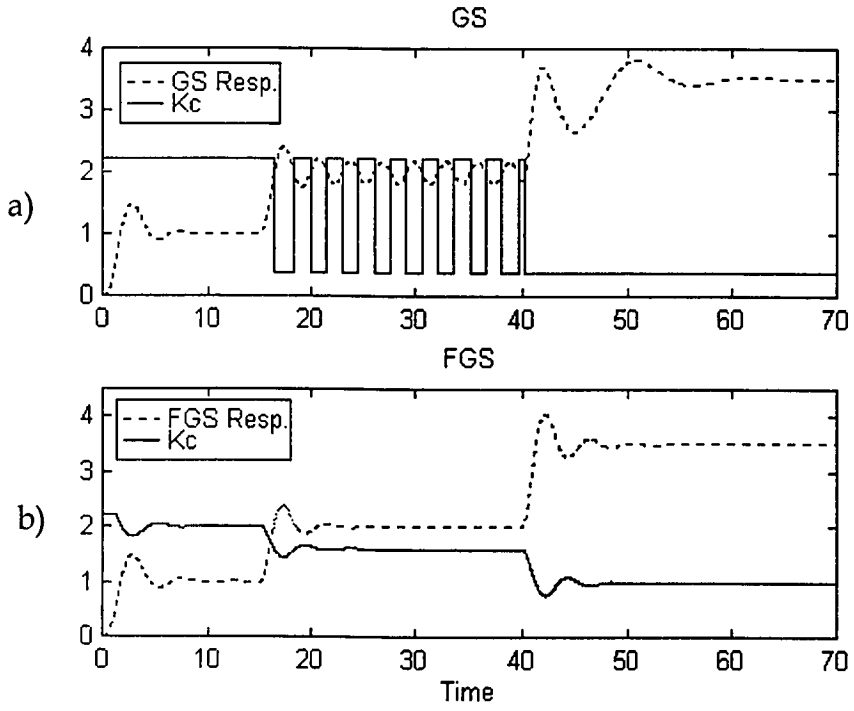


Figure 5.6 : *Simulation results of example 1-1, a) process output and controller gain K_c when using GS b) process output and controller gain K_c when using FGS*

Example 1-2

The GS and the FGS algorithms have been simulated for the following second order process whose gain changes according to (5.9) (see Figure 5.7 (a)) :

$$G_2(s) = \frac{K(y)}{(s+1)(0.5s+1)} \tag{5.8}$$

$$K(y) = \begin{cases} 2 & \text{if } y \le 1 \\ 2y & \text{if } 1 < y \le 5 \\ 10 & \text{if } y > 5 \end{cases} \tag{5.9}$$

The membership functions shown in Figure 5.2 are used in this example. Two controllers are used in this example and, hence, only two membership functions are used, one for each region. The membership function μ_{Y1} gives a weighting factor of value 1 if the process is at the first operating point (i.e., process output is less than 1) and provides a value less than 1 if the process is in transition between the two regions. The same applies to the other membership function which is related to the second region (i.e., $y > 5$).

The simulation results are shown in Figure 5.7 (b). While the FGS gives an acceptable closed loop response, the GS becomes unstable at the last setpoint change.

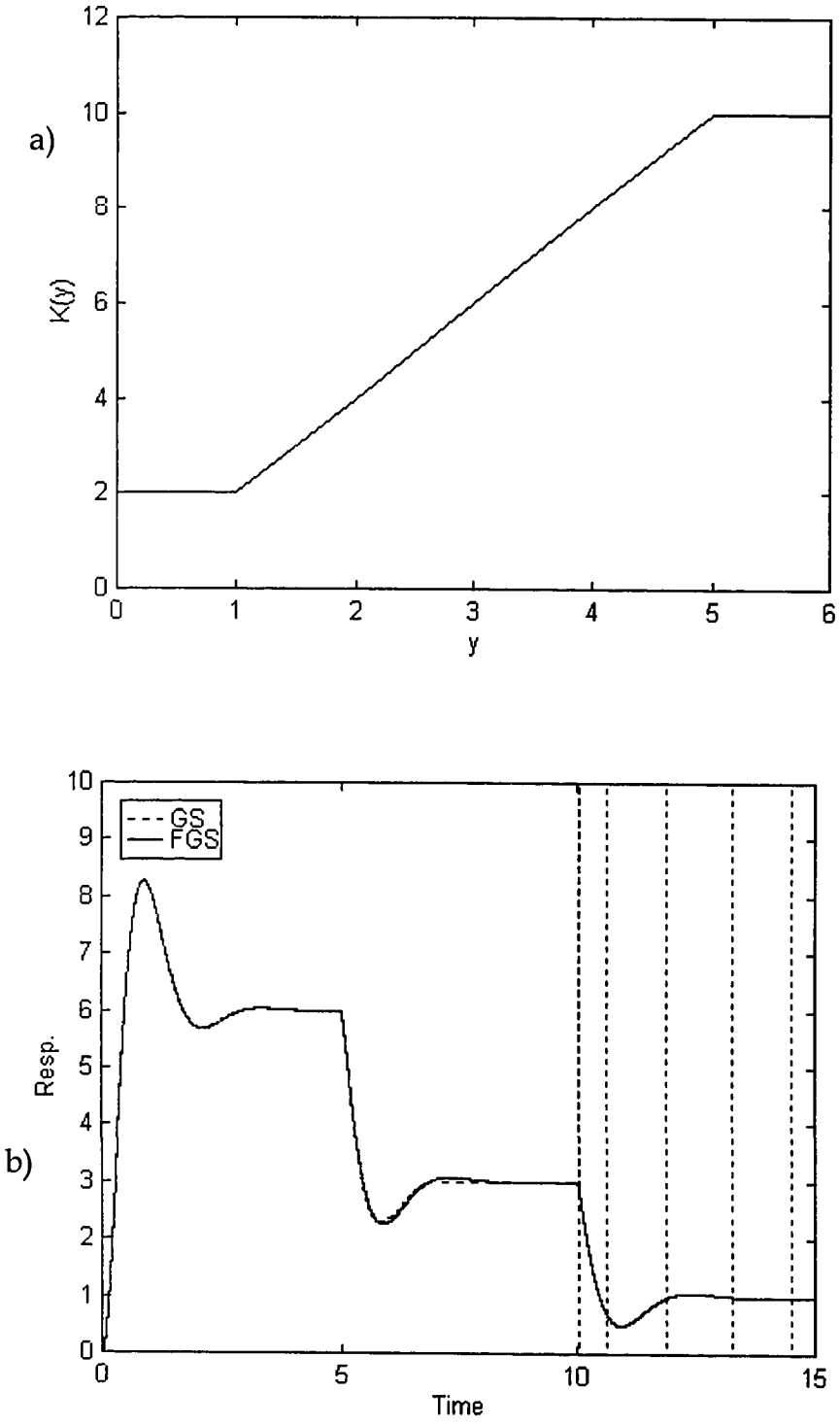


Figure 5.7 : *Simulation Results for example 1-2 , a) Process gain , b) Process outputs with GS and FGS*

Example 2-1

In this example, the performance of the proposed FL-based GS scheme is examined, and compared to that of the FGS. The process used here is the one used in example 1-2. The simulations results are shown in Figures 5.8. In Figure 5.8 (a), we see the closed loop response for both the FL-based GS scheme and the FGS and Figure 5.8 (b) shows the trajectories of α and β. Clearly, the proposed algorithm outperforms the FGS in terms of rise time, overshoot, and settling time as a result of the variations of the parameters of α and β.

Example 2-2

The next example is the following process with *K(y)* as given by (5.9) :

$$G_3(s) = \frac{K(y)\, e^{-0.4s}}{(s+1)^2} \tag{5.10}$$

The simulation results are shown in Figure 5.9. This example demonstrates, again, the improvement brought about by the proposed FL GS scheme, where the rise time is shorter, the overshoot smaller and the settling time shorter than in the FGS one. Figure 5.9 (b) shows the variation of the process gain and the controller proportional gain.

Example 2-3

Our last example is the following third order process, with *K(y)* as given by (5.9) :

$$G_4(s) = \frac{K(y)\, e^{-0.2s}}{s^3 + 2s^2 + 2.25s + 1.25} \tag{5.11}$$

Figure 5.10 shows the closed-loop response for both the FL-based GS scheme and the FGS one. While the FGS provides an oscillatory response, the proposed scheme is capable of delivering a response that is less oscillatory with smaller overshoots and undershoots.

The ISE and the IAE performance indices have been calculated for examples 2-1, 2-2, and 2-3, and are presented in Table 5.1. The content of Table 5.1 shows the enhancement in performance in terms of the ISE and IAE achieved by the new scheme in these examples.

5.6 Conclusions

FL has been utilized, in this chapter, to improve the performance of the classical GS algorithm. FL concepts have been used at the controllers' level by extending the FL-based ST algorithm presented earlier into the FL-based GS algorithm. In addition, we have used a FL inference mechanism to improve the transition between the linear controllers and to activate the proper local controller by examining the process output. The simulations results showed, in general, that the FGS outperformed the conventional GS, and that the combination of FGS and the FL-based ST algorithm can provide marked improvement over the FGS.

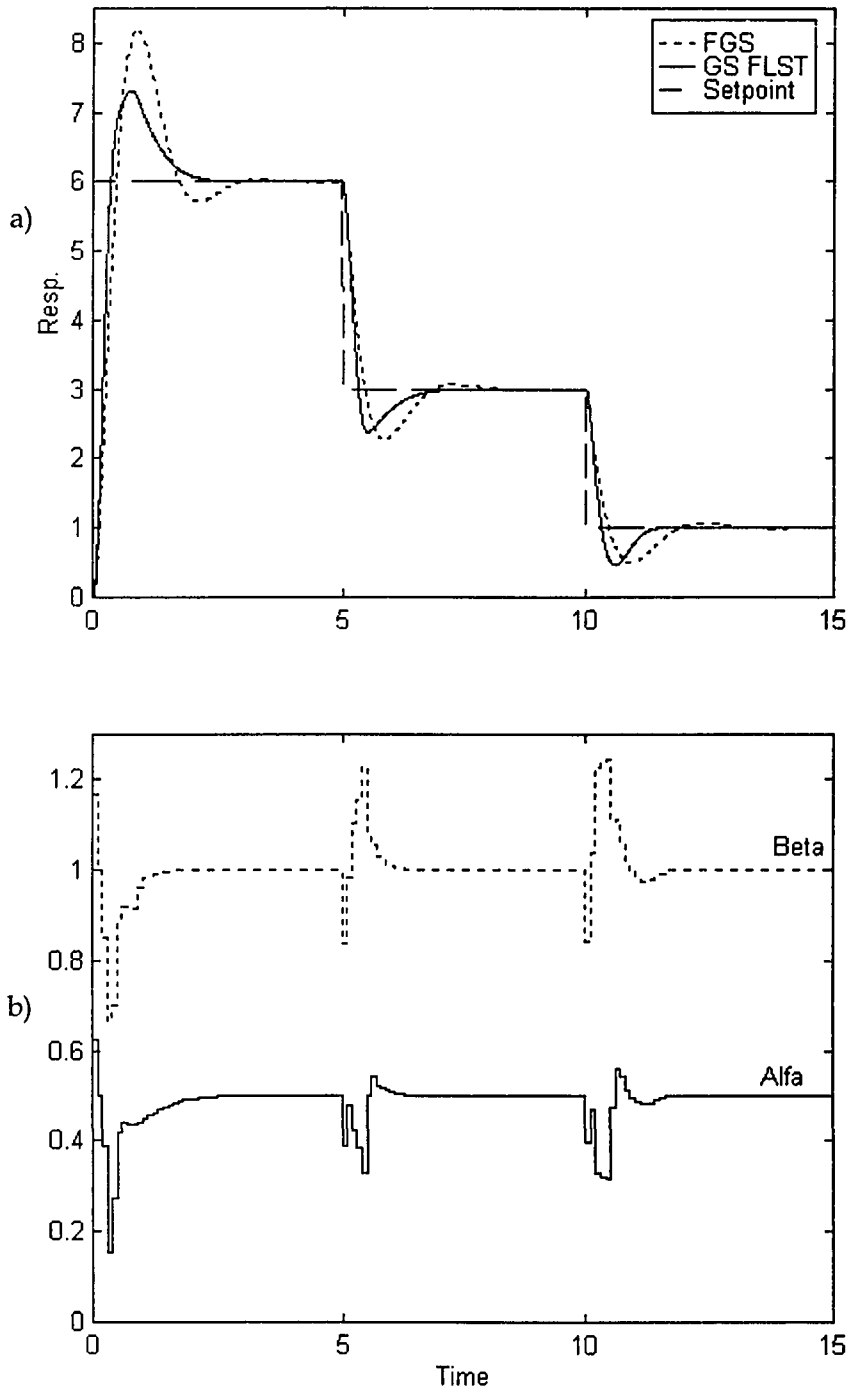
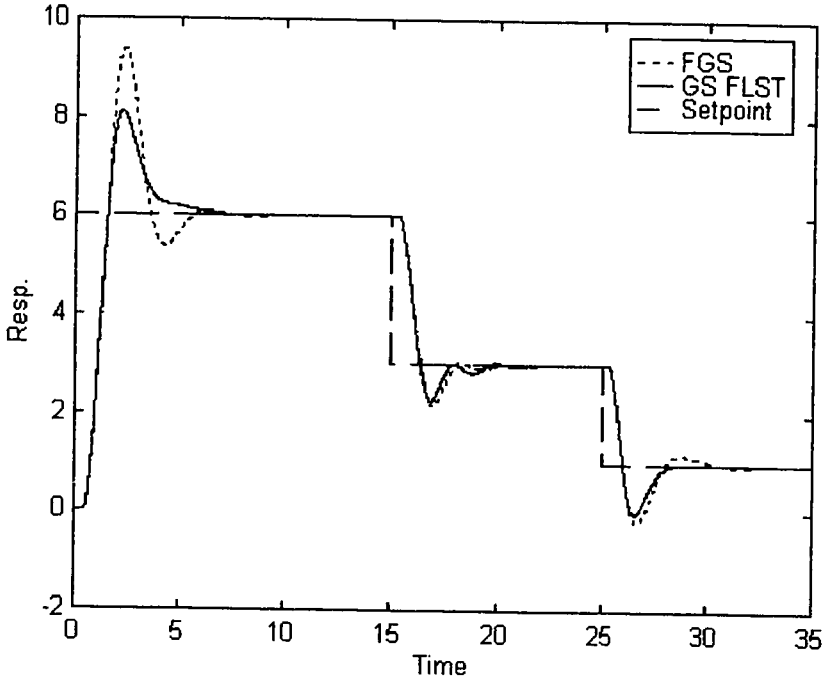


Figure 5.8 : *Simulation results for example 2-1: a) closed loop response, b)* α *and* β

a)



b)

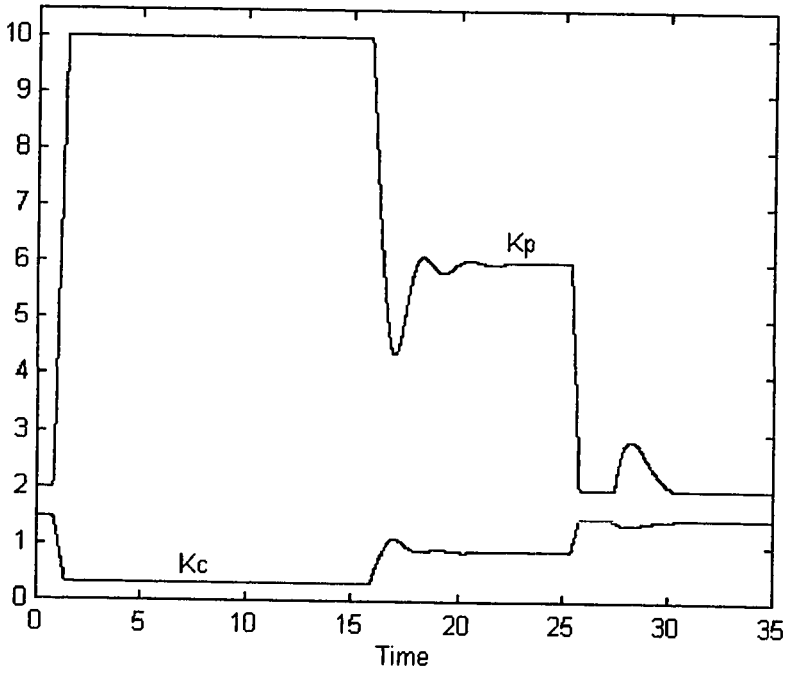


Figure 5.9 : *Simulation results for example 2-2, a) closed loop response, b) Process gain K_P and controller gain K_c*

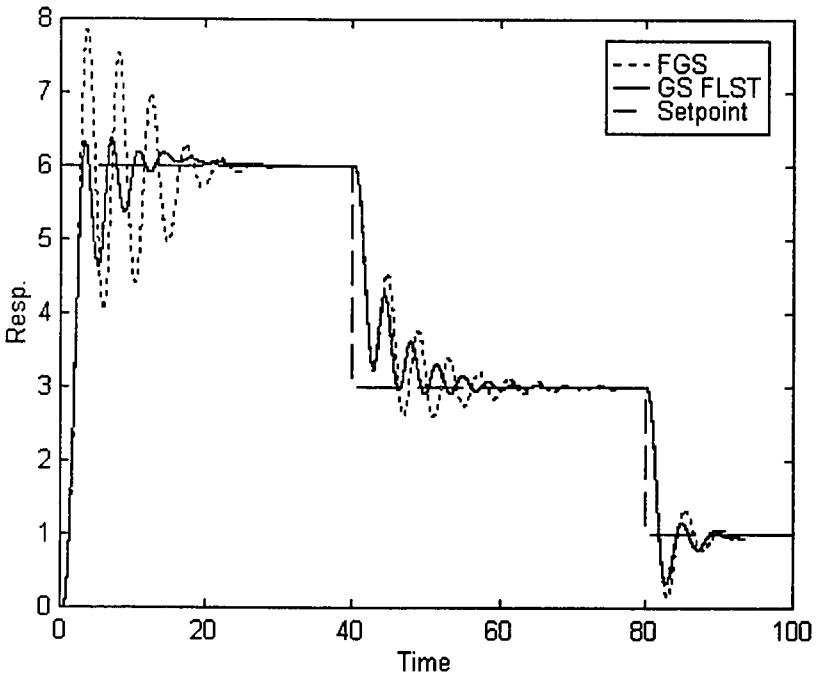


Figure 5.10 : *Simulation results for example 2-3*

Table 5.1 : *Summery of simulation results*

	FGS		FL-based GS	
	ISE	IAE	ISE	IAE
Example 2-1	12.0538	5.5151	9.2762	4.2579
Example 2-2	55.3858	18.3389	47.8133	15.9803
Example 2-3	93.4842	42.2303	76.5204	29.4098

CHAPTER 6

CONCLUSIONS

This chapter presents the summary and conclusions of this research and provides a number of recommendations for future work.

6.1 Summary and Conclusions

It is anticipated that PID controllers will continue to play a key role in process control for a long time to come. Therefore, we attempted, in this work, to look at various areas related to PID control with a view to enhancing its performance and increasing its capabilities by combining various conventional control techniques with the evolving FLC methodologies.

The general framework observed throughout this thesis can be summarized as follows. First, we have tried to combine FL inference techniques with classical feedback control strategies such as those based on PID, GS, and SP so that we augment the capabilities of these techniques

and improve their overall control. Secondly, we attempted to use, as much as possible, tuning techniques which are, from a practical perspective, effective and easy to design and implement.

In this thesis, we looked at various applications of the FL-based ST scheme, which is outlined and presented in chapter 3, to a wide range of processes. First, in chapter 3, we applied the scheme to processes with a short dead time. Then, in chapter 4, we combined the SP dead time compensation scheme with the FL-based ST algorithm to control processes with large time delays. Finally, in chapter 5, we looked into the application of FL concepts to control processes with variable gains. Here, we first investigated the FGS method proposed in the literature which utilizes FL rules to improve the transition between the various regions of the GS controller and then combined it with our FL-based ST algorithm to provide an improved performance over both the GS and the FGS.

It has been shown in chapter 3 that the proposed FL-based ST scheme can produce marked improvements in the performance of the PID controller when regulating processes with small time delay. The results of the simulation conducted in chapter 4 have shown that the combination of SP and the FL-based ST algorithm can provide enhanced performance over the original SP and that the sensitivity of the SP to model inaccuracies can be considerably reduced. In addition, it has been shown that, in the design procedure of the FL-based time delay algorithm, a good use of the relay feedback experiment has been made to derive the model of the process and obtain the initial PI controller parameters as well. In chapter 5, it has been demonstrated through simulations that the FGS outperforms the

conventional GS, and that the combination of FGS and the FL-based ST algorithm can provide clear improvement over the FGS.

One of the shortcomings of the proposed FL-based algorithm is that including the FL inference capability into the conventional control strategies considered in this thesis loads to adding some complexity and nonlinearity into the system that makes the analysis of the stability and robustness of the overall system very difficult to carry out. Another disadvantage is that the selection of the scaling factors for the FL part of the controller is done by trial and error, which is time consuming.

6.2 Recommendations for future work

Some areas that deserve further study and research have been identified as follows :

- Actual implementation of the proposed algorithms would be a good extension to the work done in this thesis. It would be beneficial to apply the FL-based controllers designed in this thesis to some process control problems, where it is customary to use PID controllers, SP, and GS controller, and asses their practical performance.

- A possible research direction is to study, at least through extensive simulation, the stability and the robustness properties of the proposed FL-based schemes.

- Including a learning capability into the FL-based ST scheme would be very useful. Two approaches can be considered. The first one is the incorporation of a fuzzy tuner that adjust the scaling factors. The other approach, which is more involved, is to modify the fuzzy control rules. The work of Maeda [23] and Zhao [36] is a good starting point.

- The work done in this thesis is based on having a PI or a PID controller structure. It would be of interest to investigate the possibility of applying a similar approach to other classical control structure such as the lead-lag, feedforward, and the cascade control algorithms.

References

1) Astrom, K. J., "*Intelligent tuning*", IFAC Adaptive systems in control and signal processing, Grenoble, France, 1992

2) Astrom, K. J., and B. Wittenmark, *Adaptive Control*, Addison-Wesley Publication Co., 1989.

3) Astrom, K. J., and B. Wittenmark, "*On self-tuning regulators*", Automatica vol.9, pp. 185-199, 1972.

4) Astrom, K. J., J. J. Anton, and K. E. Arzen, "*Expert Control*", Automatica, vol.22, no.3, pp. 277-286, 1986.

5) Astrom, K. J., and T. Hagglund, "*A frequency domain method for automatic tuning of simple feedback loops*", Proc. of 23rd Conf. on Decision & Control, Las Vegas, NV,. pp. 299-304, Dec. (1984)

6) Astrom, K. J., and T. Hagglund, *Automatic Tuning of PID Controllers*, Instrument Society of America, 1988.

7) Astrom, K. J., and T. Hagglund, "*Automatic tuning of simple regulators with specifications on phase and amplitude margins*", Automatica, vol.20, no.5, pp. 645-651, 1984.

8) Astrom, K. J., T. Hagglund, C. Hang, and W. Ho, "*Automatic Tuning and adaptation for PID controllers - A Survey*", IFAC Adaptive systems in control and signal processing, Grenoble, France, 1992.

9) Berenji, H. R., "*Fuzzy Logic Controllers*", in L. A. Zadeh and R. Yager, editors, Introduction to Fuzzy Logic Applications in Intelligent Systems, Kluwer Academic Publishers, 1991.

10) Donoghue, J. F., "*Review of control design approaches for transport delay processes*", ISA Transaction, vol. 16, no. 2, 1977.

11) Hagglund, T., "*A Predictive PI Controller for processes with long dead times*", IEEE Control Systems, pp. 57-60, February 1992.

12) Hang, C. C., K. J. Astrom, "*Practical aspects of PID auto-tuners based on relay feedback*", IFAC Adaptive Control of chemical processes, Copenhagen, Denmark, pp. 153-158, 1988.

13) Hang, C. C., Q. G. Wang, and L. S. Cao, "*Self-Tuning smith predictors for process with long dead time*", Int. Journal of Adaptive Control and Signal Processing, vol. 9, pp. 255-270, 1995.

14) Hang, C. C. and K. K. Sin, "*Development of an intelligent self-tuning PID controller*", ISA International conference and exhibition. Houston, 1992.

15) He, Shi-Zhong, Shaohua Freng-Lan Xu, and Pei-Zhuang Wang, *"Fuzzy self-tuning of PID controllers"*, Fuzzy Sets & Systems, vol.56, pp. 37-46, 1993.

16) Ho, W. K. *"Towards Intelligent PID Control"*, Ph.D. Thesis, National University of Singapore , 1991.

17) Jamshidi, M. and C. J. Herget, Editors ,*"Recent Advances in Computer-Aided Control Systems Engg."*, Elsevier Science Publishers B. V., 1992.

18) Kim, J., *"Fuzzy precompensated PID Controller"*, IEEE transaction on control systems technology, vol.3, no.4, 1994.

19) King, P. J. and E. H. Mamdani, *"The application of fuzzy control to industrial processes"*, Automatica, vol.13, pp. 235-242, 1977.

20) Langari, R. and H. R. Berenji, *"Fuzzy Logic in control engineering"*, in D. White and D. Sofg, editor, Handbook of intelligent Control, Van Nostrand Reinhold, 1992.

21) Lee, C. C., "*Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part 1 & 2*", IEEE Transaction on Systems, vol.20, no.2, 404-435, 1990.

22) Ling, C. and T. F. Edgar, *"A new fuzzy gain algorithm for process control"*, In Proc. Am. Cont. Conf., Vol. 3, pp. 2284-2290, Chicago, 1992.

23) Maeda, M. and S. Murakami, *"A self-tuning fuzzy controller"*, Fuzzy Sets & Systems, vol.51, pp. 29-40, 1992.

24) Mamdani, E. H., and S. Assilian, *"An Experiment in Linguistic Synthesis with a fuzzy logic controller"*, Internat. J. Man-Machine Stud., vol.7, no.1, pp. 1-13, 1975.

25) Marshall, J. E., *Control of time-delay Systems*, Peter Peregrinus Ltd. UK, 1979.

26) Pedrycz, W. *Fuzzy control and fuzzy systems*, Research Studies Press Ltd., 1989.

27) Seborg, D. E., T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control*, J. W. & Sons, 1989.

28) Schleck, J. R., and D. Hanesian, *"An evaluation of the Smith Linear Predictor technique for controlling dead time dominant processes"*, ISA Transaction, vol. 17, no. 4, 1978.

29) Shinskey, F. G., *Process Control Systems, Application, Design, and Tuning*, McGraw-Hill, NY, 1988.

30) Sugeno, M. *"An Introductory Survey of Fuzzy Control," Information Science*, vol.36, pp. 59-83, 1985.

31) Terano, T., K. Asai, M. Sugeno, *Fuzzy Systems Theory, and its applications*, Academic Press, Inc. San Diego, CA, 1992.

32) Tzafestas, S., and N. Papanikolopoulos, *"Incremental Fuzzy Expert PID control"*, IEEE Tans. on Industrial Electronics, vol.37, no.5, pp. 365-371, 1990.

33) Yamakawa, T., "*A fuzzy logic controller*", Journal of Biotechnology, vol.24, pp. 1-32, 1992.

34) Zadeh, L.A., "*Outline of a New Approach to the analysis of Complex Systems and Decision Processes*," IEEE Trans. on Systems, Man, and Cybernetics, vol.3, no.1, 28-44, 1973.

35) Zhao, Z. "*Fuzzy gain scheduling of PID*", IEEE Transaction on Systems, Man, and Cybernetics, vol.23, no.5, pp. 1392-1398, 1993.

36) Zhao, Z. "A *Fuzzy tuner for fuzzy logic control*", In Proc. Am. Cont. Conf., Vol. 3, pp. 2268-2272, Chicago, 1992.

APPENDIX A

SOFTWARE PROGRAMS

The MATLAB and the SIMULINK packages have been used in the simulations conducted in this thesis. In this appendix, a brief description of the main MATLAB programs used in this thesis is given. In almost all the simulations done, a main Matlab program that interacts with several Simulink block diagrams and functions is used. The programs presented in this appendix are in three parts. In Parts A, B, and C, the main Matlab programs and Simulink block diagrams used for the simulations examples in Chapters three, four, and five respectively are presented.

The main program in Part A is PID_FLST.M. It interacts with the block diagrams RELAY, PID_LOP, and FLST (see Figures A.1, A.2, A.3, and A.4). The function RLY_TST.M is called from the main program to get the initial Z-N parameters by applying the relay test. The function UPD2.M is used within the FLST block diagram in the UPDATE block as an S-Function. The function DU_COG.M is called by UPD2.M to update the values of α and β using fuzzy inference. The functions MSV.M and F_RANG.M are needed in DU_COG.M.

The main program in Part B is PI_SMITH.M. It interacts with the block diagrams SMITH, and FLSMTH (see Figures A.5 and A.6). The program

MOD_EXT.M is used initially to get the process model parameters which are then used in the main program. The function UPD2.M is used also within the FLSMTH block diagram as an S-Function. The function DU_COG.M is called, again, by UPD2.M to update the values of α and β using fuzzy inference.

The main program in Part C is PID_FLGS.M which interacts with the SIMULINK block diagrams called GS_OUR and GS_TD (see A.7 - A.12). GS_OUR represents the FGS loop to be simulated and GS_TD represent the FL-based GS loop. The functions VAR_PG.M and VAR_SFKT.M are used also within the FLSMTH block diagram as an S-Function. The function UPD2.M is used also within the GS_TD block diagram as an S-Functions. The function DU_COG.M is called, again, by UPD2.M to update the values of α and β.

The programs can be used for different processes from the ones specified by providing the process specification such as the transfer function, the simulation requirements such as the simulation time, and the initial conditions in the process specification section.

Programs Listings

Part A

```
%                *** PID_FLST.M ***
%
%     This program interacts with the SIMULAB block diagrams
%     called "RELAY", "PID_LOP", and "FLST" which represent
%     block diagrams to be simulated.  The program enables
%     the user to do the following using a menu :
%
%     1.   Intialization using Rely Feedback
%     2.   Simulation using Z-N parameters
%     3.   Run FL-Based ST
%     4.   Quit
%
%
%     Process to be simulated and simulation requirement
%
d=[1 3 3 1]; K=1; td=.4; beta=1;
sp = 1; dist = 0.5; dist_tm = 20; sim_tm = 40;
%
ts=0.01;                        % Error Sampling time
rel_amp=.01; hyst=.001;         % Relay specs
%
echo off
%
%     Main Loop
%
while 1
%
      o=menu('FL BASED PID SELF-TUNING'...
      ,'Intialization using Rely Feedback'...
      ,'Simulation using Z-N parameters'...
      ,'Run FL Auto-tuner','Quit');
%
      if o == 1;                    % Conduct the relay test
          [Ku,Tu,Kp,Ti,Td,Ki,Kd] = rly_tst(rel_amp,hyst,d,K,td);
```

```
        elseif o == 2                   %  Z-N
              x0=[0 0 0];
              [ts,x] = rk23('pid_lop',sim_tm,x0,[1e-2,1e-2,1e-2]);
              plot(ts,yzn)
        elseif o == 3                   %  FL-based ST
              x0=[0 0 0];
              sfde=0.55; sfe=0.88; sfa=1.75; sfb=2.5;
              [t,x] = rk23('flst',sim_tm,x0,[1e-2,1e-2,1e-2]);
        else; break
        end
end


function [Ku,Tu,Kp,Ti,Td,Ki,Kd] = rly_tst(rel_amp,hyst,d,K,td)
%
%       This function finds the Z-N PID tuning parameters
%
Kp=0;Ki=0;Kd=0; x0=[0 0 0 0];
%
%       Bring process into LC and determine process characteristics
%
t = rk23('relay',15,x0,[1e-3,1e-3,1e-2]);
[n,m] = size(t)
upt = t((n-700):n); upyout =y1((n-700):n);
amp = (abs(max(upyout)) + abs(min(upyout)))/2;
st = size(upt); j = 1;
%
clear t; clear x; clear y1;
old_sgn = sign(upyout(1));
for i = 2:st;
        sgn = sign(upyout(i));
        if (old_sgn ~= sgn);
              zro_crs(j) = (upt(i) + upt(i-1))/2;
              j=j+1;
        end;
        old_sgn = sgn;
        if j == 3; break; end
end
%
%       Compute Tu and Ku
%
Tu = 2 * (zro_crs(2) - zro_crs(1));
Ku = (4*rel_amp) / (pi * sqrt(amp*amp-hyst*hyst));
%
```

```
%       Determine PI or PID parameters
%
mod= menu('Choose Controller Type','PI','PID');
if mod == 1
        Kp = .45*Ku; Ti = .85*Tu; Ki=Kp/Ti; Kd=0;
elseif mod ==2
        Kp = .6*Ku; Ti = .5*Tu; Td = .125*Tu;
        Ki = Kp/Ti; Kd = Kp*Td;
end
plot(upt,upyout)


function [sys,x0] = upd2(t, x, u, flag)
%
%       This function updates the values of α and β using the function
%       du_cog.m.  This function is used within an S-Function in
%       Simulink.
%
flag = abs(flag);
offset = 0;
ts = 0.2;                       %  FL scheme sampling time
%
if (flag == 3)
%
%       Is it a sample hit (within a tolerance of 1e-6) ?
%       If it is not a discrete sample hit, the output
%       should not change, so just return
%       previous Alfa and beta.
%
        if abs(round((t-offset)/ts)-(t-offset)/ts) < 1e-6
              h=du_cog(u(1),u(2),1);
              sys(1)=h;                         % Alfa
              h=du_cog(u(1),u(2),0);
              sys(2)=h;                         % Beta
               [t u(1) u(2) sys(1) sys(2)]
        end
%
elseif flag == 4          %  Return next sample hit
%
%       ns stores the number of samples
        ns = (t-offset)/ts;
%       This is the time of the next sample hit.
        sys = offset + (1+floor(ns+1e-13*(1+ns)))*ts;
```

```
%
elseif (flag == 0)
        sys(1) = 0;         % number continuous states
        sys(2) = 0;         % number discrete states
        sys(3) = 2;         % number of outputs
        sys(4) = 2;         % number of inputs
        sys(5) = 0;         % unused
        sys(6) = 0;         % has direct feedthrough
        x0 = [];
else
        sys = [];
end


function h = du_cog(e,de,swtch)
%
%       This function calculates the defuzzified output
%       based on certain (e) and (de).
%
%       Fuzzy logic rules matrix
%
if swtch == 1;
%
% (tab. 1.4, alfa)
        flr = [1 1 2 7 6 5 4;1 2 3 6 5 4 3;1 2 3 5 4 3 2;1 2 3 4 3 2 1];
        flr = [flr;2 3 4 5 3 2 1;3 4 5 6 3 2 1;4 5 6 7 2 1 1];
else;
% (table 1.4, beta)
        flr = [1 1 1 1 2 3 4;1 1 2 2 3 4 5;1 2 3 3 4 5 6;1 2 3 4 5 6 7];
        flr = [flr;2 3 4 5 5 6 7;3 4 5 6 6 7 7;4 5 6 7 7 7 7];
end
%
%       Determine membership functions fired by (e)
%
if      ( e <= (-.75));             mse = [0 1];
elseif  (e >= (-.75) & e < (-.5));  mse = [1 2];
elseif  (e >= (-.5) & e < (-.25));  mse = [2 3];
elseif  (e >= (-.25) & e < (0.0));  mse = [3 4];
elseif  (e >= (0.0) & e < (.25));   mse = [4 5];
elseif  (e >= (.25) & e < (0.5));   mse = [5 6];
elseif  (e >= (.5) & e < (.75));    mse = [6 7];
elseif  (e >= (.75));               mse = [7 8];
end
%
```

```
%       Determine membership functions fired by (de)
%
if      (de <= (-.75));                 msde = [0 1];
elseif  (de >= (-.75) & de < (-.5));    msde = [1 2];
elseif  (de >= (-.5) & de < (-.25));    msde = [2 3];
elseif  (de >= (-.25) & de < (0.0));    msde = [3 4];
elseif  (de >= (0.0) & de < (.25));     msde = [4 5];
elseif  (de >= (.25) & de < (0.5));     msde = [5 6];
elseif  (de >= (.5) & de < (.75));      msde = [6 7];
elseif  (de >= (.75));                  msde = [7 8];
end
%
%       Determine (du) membership functions caused
%       by (e) and (de) and the values at these functions
%
sci=1; eci=2; scj=1; ecj=2;
if (mse(2) == 1); sci=2; end
if (mse(1) == 7); eci=1; end
if (msde(2) == 1); scj=2; end
if (msde(1) == 7); ecj=1; end
%
msdu = zeros(2,2); msduv = zeros(2,2);
ra = 1; rb = -1;
%
for i = sci:eci
  for j = scj:ecj
        msdu(i,j) = flr(mse(i),msde(j));
        [a,b] = f_rang(msdu(i,j));
        ra = min(ra,a); rb = max(rb,b);
        [a,b] = f_rang(mse(i));
        msev = msv(e,a,b,1);            % Evaluate ms function at e
        [a,b] = f_rang(msde(j));
        msdev = msv(de,a,b,1);          % Evaluate ms function at de
        msduv(i,j) = min(msev,msdev);
  end
end
%
%       Calculate Center Of Gravity (COG)
%
den_cog = 0.0; num_cog = 0.0; val = 0.0;
q=1;
for k = ra:0.01:rb
        for i = sci:eci
              for j = scj:ecj
```

```
                [a,b] = f_rang(msdu(i,j));
                v = msv(k,a,b,msduv(i,j));
                val = max(val,v);
            end
        end
        l(q)=val;
        q=q+1;
        den_cog = den_cog + val;
        num_cog = k*val + num_cog;
        val = 0.0;
end
%
cog = num_cog / den_cog;
h = cog;
```

```
function [a,b] = f_rang(msf)
%
% This function returns the starting point (a) and
% the ending point (b) of the membership function (msf).
%
if      (msf == 1); a = -1;    b = -.5;     % NL membership fun.
elseif (msf == 2); a = -.75;  b = -.25;    % NM membership fun.
elseif (msf == 3); a = -.5;    b = 0.0;     % NS membership fun.
elseif (msf == 4); a = -.25;  b =.25;      % Z  membership fun.
elseif (msf == 5); a = 0.0;    b = .5;      % PS membership fun.
elseif (msf == 6); a = .25;    b = .75;     % PM membership fun.
elseif (msf == 7); a = .5;     b = 1;       % PL membership fun.
end;
```

```
function y = msv(x,a,b,c)
%
% This function evaluates a membership function whose
% dimensions are a, b, and c at the point x.
%
mid_rng = (b-a)/2;
a1 = a + c*mid_rng; b1= b - c*mid_rng;
ls = (x-a) / mid_rng; rs = (b-x) / mid_rng;
%
if (a == -1);
        if(x <= b1); y = c;
        elseif (x > b1 & x <= b); y = rs;
        else; y = 0;
```

```
        end
elseif (b == 1)
        if(x >= a & x < a1); y = ls;
        elseif ( x >= a1); y = c;
        else; y = 0;
        end
else
        if (x > a & x <= a1);  y = ls;
        elseif (x > a1 & x <= b1); y = c;
        elseif (x > b1 & x <= b); y = rs;
        else  y = 0;
        end
end
```

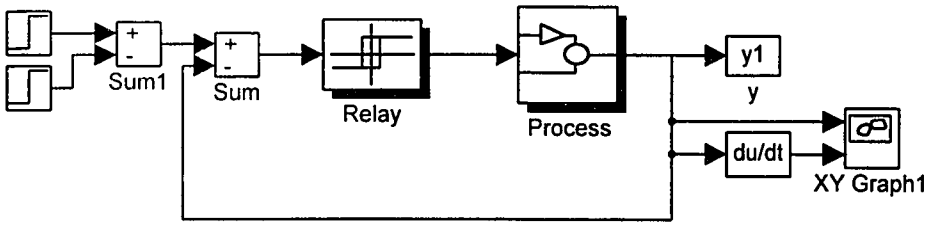


Figure A.1 : *RELAY*

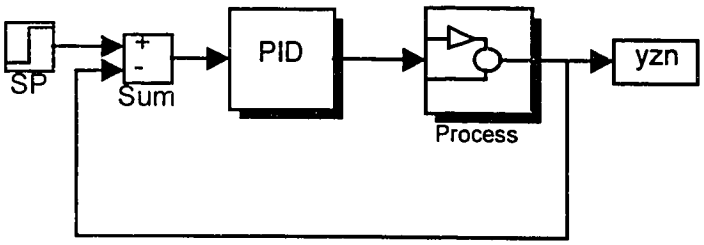


Figure A.2 : *PID_LOP*

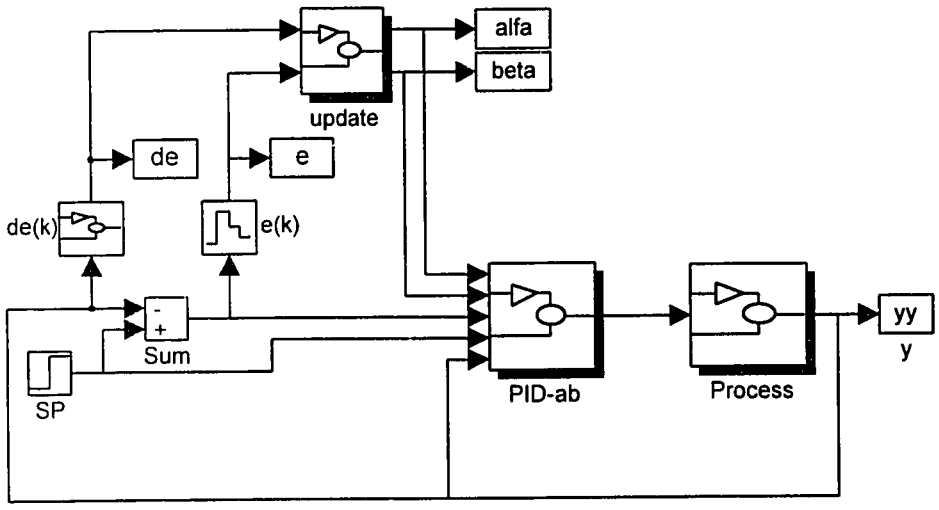
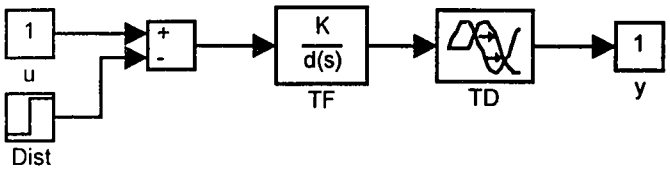
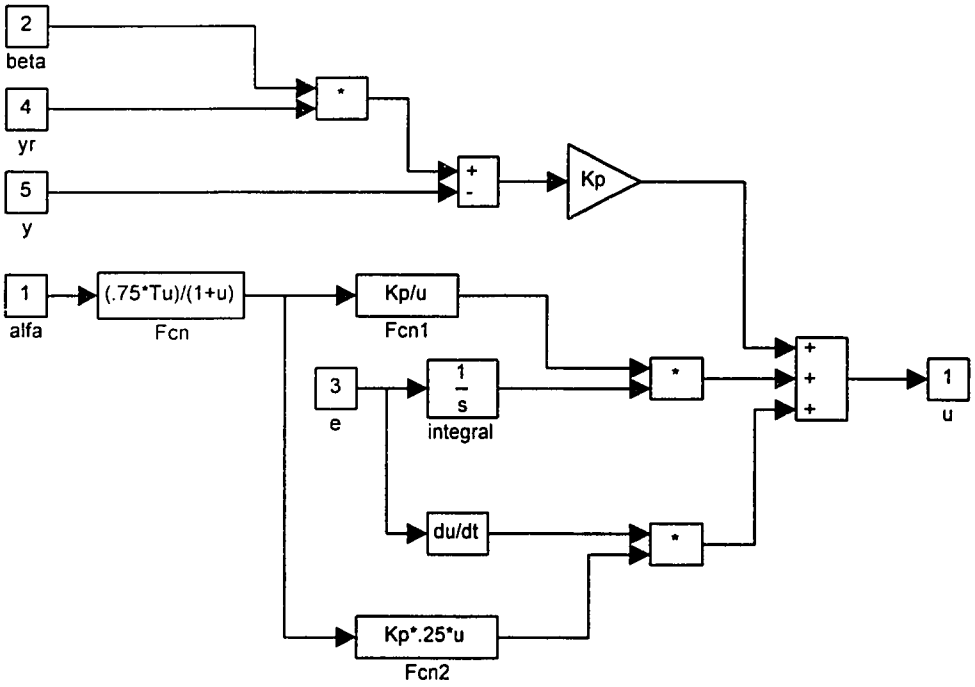


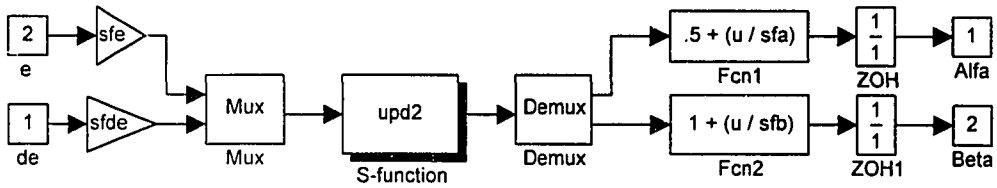
Figure A.3 : *FLST*



a)



b)



c)

Figure A.4 : *a) Process b) PID controller c) α and β update block*

Part B

```
%
%                *** PI_SMITH.M ***
%
%     This program interacts with the SIMULINK block
%     diagrams called "SMITH", "FLSMTH" which represent
%     the block diagram of the loop to be simulated.
%
%     The program enables the user to do the
%     following using a menu :
%
%     1.   Simulation using Z-N parameters
%     2.   Simulation using SP
%     3.   Simulation using FL-Based SP
%     4.   Quit
%
%
%     Process to be simulated, SP model and simulation
%     requirement
%
K=.57; td=18.7; d=conv([8.6 1],[8.6 1]);                  % Process
tdm=18.72; dm=conv([8.49 1],[8.49 1]);                        % Model
sp = 1; dist = 0; dist_tm = 0; sim_tm=100; beta=1;            % Sim. Req.
Ku=20.08 ;Tu=18.78 ;Kp=.45*Ku; Ti=.85*Tu;Ki=Kp/Ti;Kd=0;  % controller
%
ts=0.01;          % Error Sampling time
%
echo off
%
%     Main loop
%
while 1
%
      o=menu('FL-BASED TIME DELAY COMPESATION'...
      ,'Simulation using Z-N parameters'...
      ,'Smith','FL-BASED Sp','Quit');

      if o == 1                          % Z-N Simulation
```

```
            x0=[0 0]; td=0; d=dm;
            ts = rk23('pid_lop',sim_tm,x0,[1e-2,1e-2,1e-2]);
%
        elseif o == 2                       % Smith
            x0=[0 0 0 0 0];
            tsm = rk23('smith',sim_tm,x0,[1e-2,1e-2,1e-2]);
            plot(tsm,ysm)
%
        elseif o == 3                       % FL Smith
            x0=[0 0 0 0 0 1];
            tsmf = rk23('flsmth',sim_tm,x0,[1e-2,1e-2,1e-2]);
%
        else; break
        end
end


%
%                *** MOD_EXT.M ***
%
%       This program finds the first or the second order
%       model parameters (See section 4.3 for more details)
%
%
model= menu('Choose Model Type','First order + DT'...
,'Second order + DT');
if model == 1
        tw1 = (Tu/2/pi) * sqrt(Ku*Ku*K*K-1)
        L1 = (Tu/2/pi) * (pi - atan(2*pi*tw1/Tu))
elseif model ==2
        tw2 = (Tu/2/pi) * sqrt(Ku*K-1)
        L2 = (Tu/2/pi) * (pi - 2*atan(2*pi*tw2/Tu))
end
```

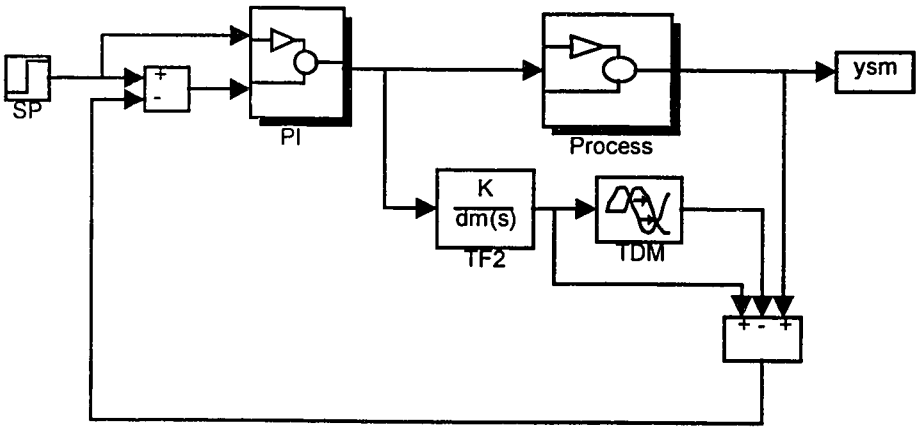


Figure A.5 : *SMITH*

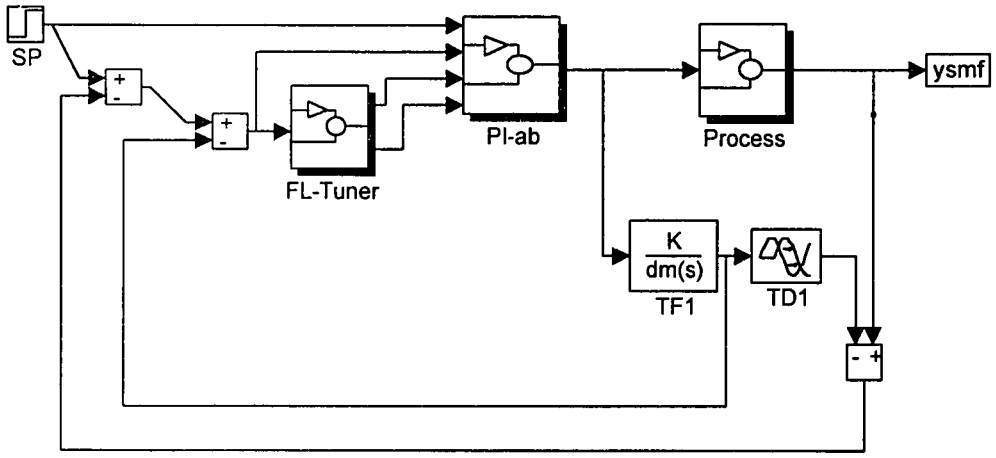


Figure A.6 : *FLSMTH*

Part C

```
%
%                *** PID_FLGS.M ***
%
%     This program interacts with the SIMULINK block
%     diagrams called "GS_OUR", "GS_TD" which represent
%     the block diagram of the loop to be simulated.
%
%     The program enables the user to do the
%     following using a menu :
%
%     1.   Simulation using FGS
%     2.   Simulation using FL-based GS
%     4.   Quit
%
%     Process to be simulated and simulation requirement
%
d=conv([1 1],[0.5 1]); td=0;
sp=1;dist=0; dist_tm=0; sim_tm=16; beta=1;
%
ts=0.01;          % Error Sampling time
%
global Ku1 Tu1 Ku2 Tu2
global Kp1 Ki1 Kd1 Kp2 Ki2 Kd2
%
%     Local Controllers parameters
%
Ku1=11.5201; Tu1=1.0056;          % K=2
Ku2=2.9106; Tu2=0.8789;           % K=10
%
Kp1=.6*Ku1; Ti1=.5*Tu1; Td1=.125*Tu1;
Ki1=Kp1/Ti1; Kd1=Kp1*Td1;
Kp2=.6*Ku2; Ti2=.5*Tu2; Td2=.125*Tu2;
Ki2=Kp2/Ti2; Kd2=Kp2*Td2;
%
echo off
%
%     Main loop
```

```
%
while 1
%
        o=menu('FL-BASED PID SELF-TUNING'...
        ,'GS','GS with FL Auto-tuner','Quit');
%
        if o == 1                % FGS
              x0=[0 0 0 0];
              tgs = rk23('gs_our',sim_tm,x0,[1e-2,1e-2,1e-2]);
              plot(tgs,ygs)
        elseif o == 2            % FL-Based GS
              x0=[0 0 0 0];
              tt = rk23('gs_td',sim_tm,x0,[1e-2,1e-2,1e-2]);
        else; break
        end
end


function [sys,x0] = var_pg(t, x, u, flag)
%
%       This function provides the variable process gain.
%       This function is used within an S-Function in the
%       Simulink block diagram "GS-TD".
%
flag = abs(flag);
%
if (flag == 3)
        if(u<=1); sys=2;
        elseif(u<=5 & u>1); sys=2*u;
        else; sys=10;
        end;
elseif (flag == 0)
        sys(1) = 0;         % no continuous states
        sys(2) = 0;         % no discrete states
        sys(3) = 1;         % number of outputs
        sys(4) = 1;         % number of inputs
        sys(5) = 0;         % unused
        sys(6) = 0;         % has direct feedthrough
        x0 = [];
else
        sys = [];
end
```

```
function [sys,x0] = var_sfkt(t, x, u, flag)
%
%       This function interpolates between the local controllers
%       gains using FGS. This function is used within an S-
%       Function in the Simulink block diagram "GS-TD" and
%       "GS-OUR"
%
global Ku1 Tu1 Ku2 Tu2
%
d=conv([1 1],[.5 1]); td=0
sfe1=0.2; sfde1=15; sf_alfa1=1.5; sf_beta1=5;
sfe2=0.5; sfde2=7; sf_alfa2=1.75; sf_beta2=4;
%
flag = abs(flag);
%
if (flag == 3)
%
% FGS
%
  if(u<=1);
%
        sys(1)=Ku1; sys(2)=Tu1;
        sys(5)=sfe1; sys(6)=sfde1;
        sys(3)=sf_alfa1; sys(4)=sf_beta1;
  elseif (u<=5 & u>1);
        w1=(5-u)/4; w2=(u-1)/4;
        sys(1)=(w1*Ku1)+(w2*Ku2);
        sys(2)=(w1*Tu1)+(w2*Tu2);
        sys(5)=(w1*sfe1)+(w2*sfe2);
        sys(6)=(w1*sfde1)+(w2*sfde2);
        sys(3)=(w1*sf_alfa1)+(w2*sf_alfa2);
        sys(4)=(w1*sf_beta1)+(w2*sf_beta2);
  else;
        sys(1)=Ku2; sys(2)=Tu2;
        sys(5)=sfe2; sys(6)=sfde2;
        sys(3)=sf_alfa2; sys(4)=sf_beta2;
  end
%
elseif (flag == 0)

  sys(1) = 0;         % no continuous states
  sys(2) = 0;         % no discrete states
  sys(3) = 6;         % number of outputs
  sys(4) = 1;         % number of inputs
```

```
  sys(5) = 0;          % unused
  sys(6) = 0;          % has direct feedthrough
  x0 = [];
else
  sys = [];
end
```

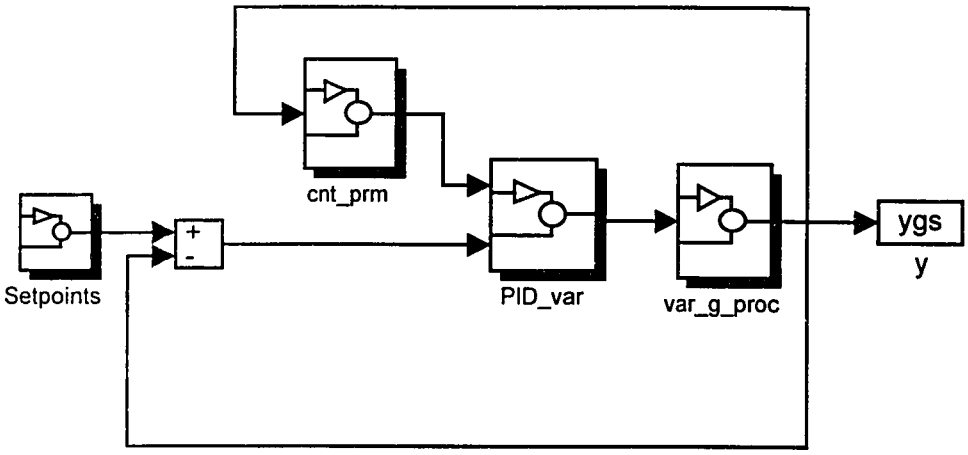
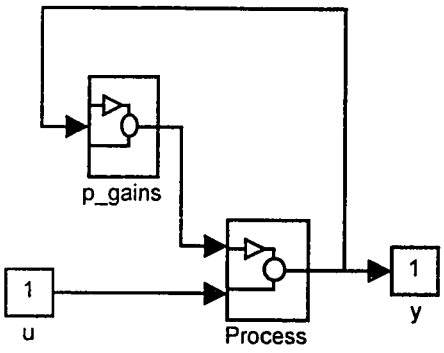


Figure A.7 : *GS_OUR.*

a)



b)

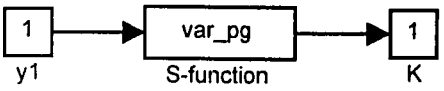
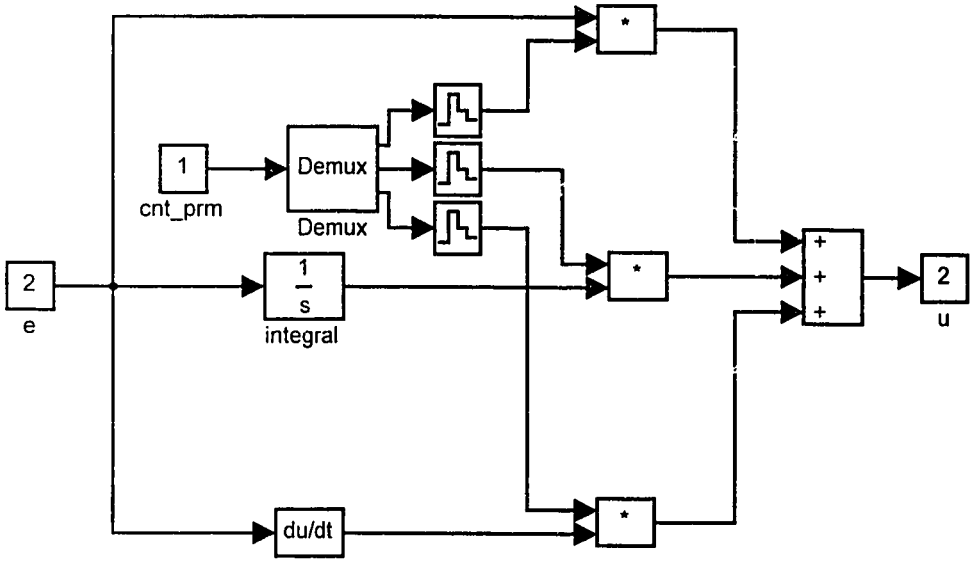
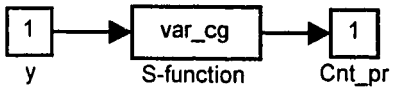


Figure A.8 : *a) Var. gain process b) Process gains (p_gains) block*



a)



b)

Figure A.9 : *a) Variables gains PID(PID-var) block b) Controller Parameters Block (cnt_prm)*

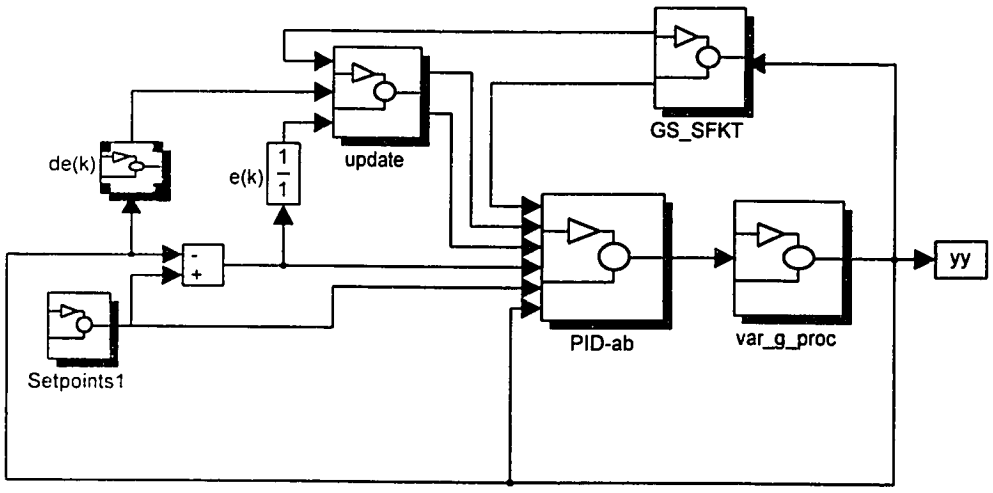
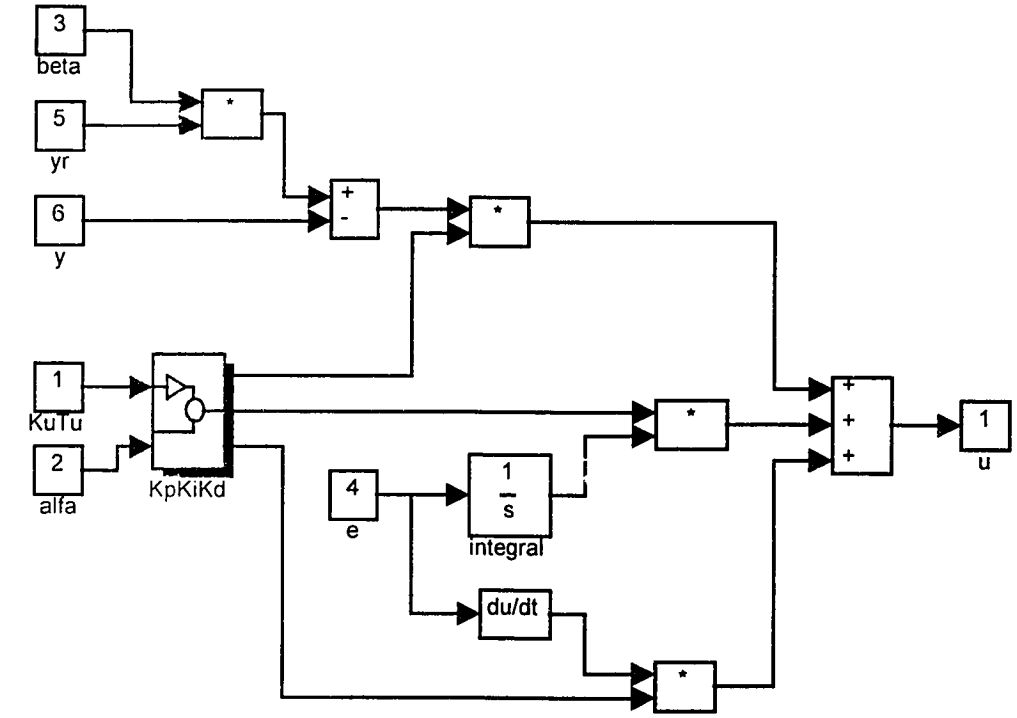
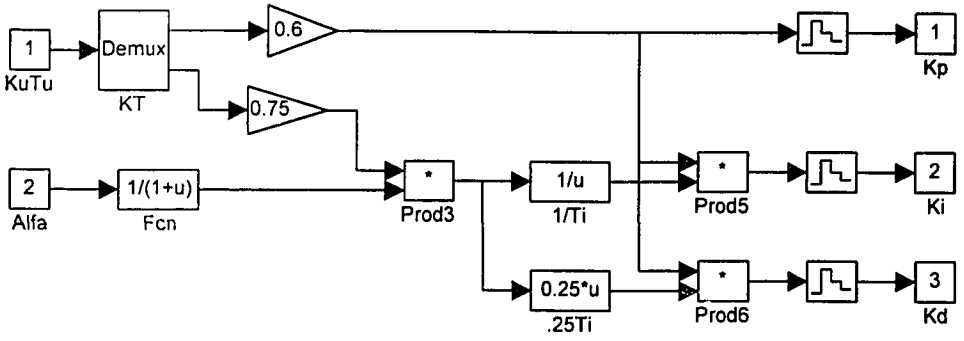


Figure A.10 : *GS-TD*



a)



b)

Figure A.11 : *a) PID-ab block b) KpKiKd block*

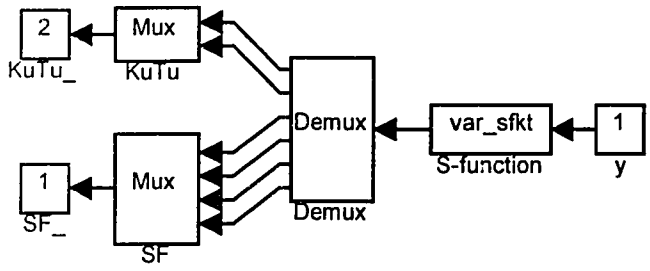
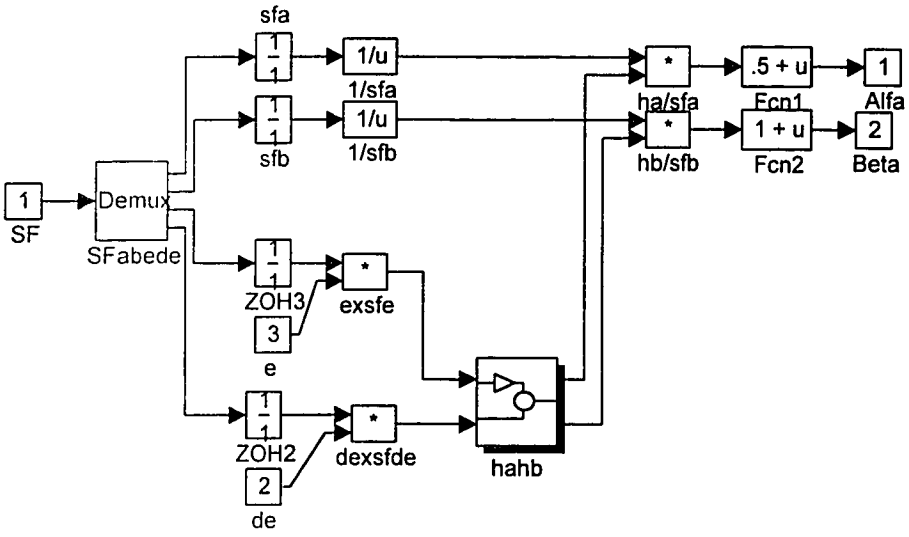
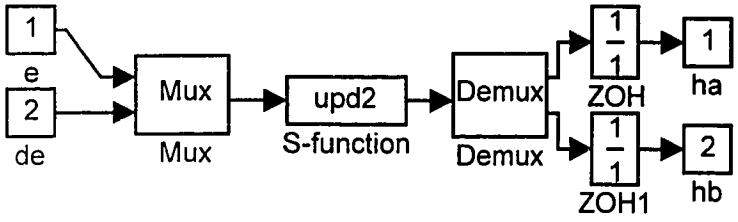


Figure A.12 : *GS_SFKT*



a)



b)

Figure A.13 : *a) update block b) hahb block*

Vita

- Taher M. Al-Nemer
- Born in Dammam, Saudi Arabia
- Received Bachelor's degree in Systems Engineering from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia in January 1990.
- Worked for the Saudi Arabian Oil Company (Saudi Aramco) for four years.
- Completed Master's degree requirements at KFUPM, Dhahran, Saudi Arabia in March 1996